```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPPP
UUU         UUU  EEE                     TTT        PPP         PPP
UUU         UUU  EEE                     TTT        PPP         PPP
UUU         UUU  EEE                     TTT        PPP         PPP
UUU         UUU  EEE                     TTT        PPP         PPP
UUU         UUU  EEE                     TTT        PPP         PPP
UUU         UUU  EEE                     TTT        PPP         PPP
UUU         UUU  EEEEEEEEEEEE            TTT        PPPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE            TTT        PPPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE            TTT        PPPPPPPPPPPPP
UUU         UUU  EEE                     TTT        PPP
UUU         UUU  EEE                     TTT        PPP
UUU         UUU  EEE                     TTT        PPP
UUU         UUU  EEE                     TTT        PPP
UUU         UUU  EEE                     TTT        PPP
UUU         UUU  EEE                     TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE         TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE         TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE         TTT        PPP
```

FILEID**UETCOMS00

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC    OOOOOO    MM      MM   SSSSSSSS    OOOOOO      OOOOOO
UU      UU  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC    OOOOOO    MM      MM   SSSSSSSS    OOOOOO      OOOOOO
UU      UU  EE              TT      CC          OO      OO  MMMM  MMMM  SS          OO      OO  OO      OO
UU      UU  EE              TT      CC          OO      OO  MMMM  MMMM  SS          OO      OO  OO      OO
UU      UU  EE              TT      CC          OO      OO  MM  MM  MM  SS          OO    OOOO  OO    OOOO
UU      UU  EE              TT      CC          OO      OO  MM  MM  MM  SS          OO    OOOO  OO    OOOO
UU      UU  EEEEEEEE        TT      CC          OO      OO  MM      MM    SSSSSS    OO  OO  OO  OO  OO  OO
UU      UU  EEEEEEEE        TT      CC          OO      OO  MM      MM    SSSSSS    OO  OO  OO  OO  OO  OO
UU      UU  EE              TT      CC          OO      OO  MM      MM          SS  OOOO    OO  OOOO    OO
UU      UU  EE              TT      CC          OO      OO  MM      MM          SS  OOOO    OO  OOOO    OO
UU      UU  EE              TT      CC          OO      OO  MM      MM          SS  OO      OO  OO      OO
UU      UU  EE              TT      CC          OO      OO  MM      MM          SS  OO      OO  OO      OO      ....
UUUUUUUUUU  EEEEEEEEEE      TT        CCCCCCCC    OOOOOO    MM      MM   SSSSSSSS    OOOOOO      OOOOOO        ....
UUUUUUUUUU  EEEEEEEEEE      TT        CCCCCCCC    OOOOOO    MM      MM   SSSSSSSS    OOOOOO      OOOOOO        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE UETCOMS00 VAX/VMS UETP DEVICE TEST FOR DMC/DMR
0000     2          .IDENT  'V04-000'
0000     3          .ENABLE SUPPRESSION
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                              *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                              *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:
0000    31  ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    32  ;       account.
0000    33  ;
0000    34  ; ABSTRACT:
0000    35  ;       This is the test program for DMC 11 / DMR 11 UETP device test
0000    36  ;
0000    37  ; ENVIRONMENT:
0000    38  ;       This program will run in user access mode, with AST enabled except
0000    39  ;       during error processing. This program requires the following
0000    40  ;       privileges and  quotas:
0000    41  ;
0000    42  ;--
0000    43  ;
0000    44  ; AUTHOR: Paul Jenq,    CREATION DATE: May, 1981
0000    45  ;
0000    46  ; MODIFIED BY:
0000    47  ;
0000    48  ;       V03-008 RNH0007         Richard N. Holstein,    15-Feb-1984
0000    49  ;               Take advantage of new UETP message codes.  Fix SSERROR
0000    50  ;               interaction with RMS_ERROR.
0000    51  ;
0000    52  ;       V03-007 RNH0006         Richard N. Holstein,    19-Dec-1983
0000    53  ;               Give correct sentinels to Test Controller.
0000    54  ;
0000    55  ;       V03-006 RNH0005         Richard N. Holstein,    07-Dec-1983
0000    56  ;               Fix bug causing attention AST error messages.
0000    57  ;
```

```
0000     58 ;        V03-005 RNH0004         Richard N. Holstein,    11-Nov-1983
0000     59 ;                Use decimal conversion routine for unit numbers.
0000     60 ;
0000     61 ;        V03-004 RNH0003         Richard N. Holstein,    11-Mar-1983
0000     62 ;                Don't signal ending message in EXIT_HANDLER.
0000     63 ;
0000     64 ;        V03-003 RNH0002         Richard N. Holstein,    25-Feb-1983
0000     65 ;                Allow for longer device names.  Fix error numbering bug.
0000     66 ;
0000     67 ;        V03-002 RNH0001         Richard N. Holstein,    03-Nov-1982
0000     68 ;                Miscellaneous fixes listed in the V3B UETP Workplan.
0000     69 ;
0000     70 ;        V03-001 LDJ0002         Larry D. Jones,         03-Sep-1982
0000     71 ;                Fixed LOOP mode bug causing device offline error message.
0000     72 ;**
```

```
                     0000     74          .SBTTL  Declarations
                     0000     75 ;
                     0000     76 ; INCLUDE FILES:
                     0000     77 ;
                     0000     78 ;       SYS$LIBRARY:LIB.MLB     for general definitions
                     0000     79 ;       SHRLIB$:UETP.MLB        for UETP definitions
                     0000     80
                     0000     81 ;
                     0000     82 ; MACROS:
                     0000     83 ;
                     0000     84          $CHFDEF                         ; Condition handler frame definitions
                     0000     85          $DEVDEF                         ; Device definitions
                     0000     86          $DIBDEF                         ; Device Information Block
                     0000     87          $DVIDEF                         ; $GETDVI ITMLST item codes
                     0000     88          $SHRDEF                         ; Shared messages
                     0000     89          $SSDEF                          ; System Service status codes
                     0000     90          $STSDEF                         ; Status return
                     0000     91          $UETUNTDEF                      ; UETP unit block offset definitions
                     0000     92          $UETPDEF                        ; UETP
                     0000     93          $XMDEF                          ; DMC/DMR chars and status definition
                     0000     94          $MSGDEF                         ; mailbox message type definition
                     0000     95 ;
                     0000     96 ; EQUATED SYMBOLS:
                     0000     97 ;
                     0000     98 ;   Facility number definitions:
00000001             0000     99          RMS$_FACILITY = 1
                     0000    100
                     0000    101 ;   SHR message definitions:
00740000             0000    102          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0             0000    103          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038             0000    104          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080             0000    105          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098             0000    106          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130             0000    107          UETP$_TEXT   = UETP!SHR$_TEXT
                     0000    108
                     0000    109 ;   Internal flag bits...:
00000001             0000    110          TEST_OVERV    = 1               ; Set when test is over
00000002             0000    111          SAFE_TO_UPDV  = 2               ; Set if it's safe to update UETINIDEV
00000003             0000    112          BEGIN_MSGV    = 3               ; Set if "BEGIN" msg has been printed
00000004             0000    113          MODE_IS_ONEV  = 4               ; Set when the MODE is ONE
00000005             0000    114          TEST_ERRV     = 5               ; Set when intended introduce error for  tes
00000006             0000    115          FLAG_SHUTDNV  = 6               ; Set to indicate device should be
                     0000    116                                          ; shutdown if errors occur
                     0000    117 ;   ...and corresponding masks:
00000002             0000    118          TEST_OVERM    = 1@TEST_OVERV
00000004             0000    119          SAFE_TO_UPDM  = 1@SAFE_TO_UPDV
00000008             0000    120          BEGIN_MSGM    = 1@BEGIN_MSGV
00000010             0000    121          MODE_IS_ONEM  = 1@MODE_IS_ONEV
00000020             0000    122          TEST_ERRM     = 1@TEST_ERRV
00000040             0000    123          FLAG_SHUTDNM  = 1@FLAG_SHUTDNV
                     0000    124
                     0000    125 ;   Miscellany:
00000020             0000    126          LC_BITM       = ^X20            ; Mask to convert lower case to upper
00000028             0000    127          REC_SIZE      = 40              ; UETINIDEV.DAT record size
00000084             0000    128          TEXT_BUFFER   = 132             ; Internal text buffer size
00000004             0000    129          EFN2          = 4               ; EFN used for three minute timer
00000003             0000    130          SS_SYNCH_EFN  = 3               ; Synch miscellaneous system services
```

```
0000000F  0000   131          MAX_PROC_NAME = 15                  ; Longest possible process name
0000000A  0000   132          MAX_DEV_DESIG = 10                  ; Longest possible controller name
00000005  0000   133          MAX_UNIT_DESIG= 5                   ; Longest possible unit number
00000080  0000   134          MBXSIZE       = ^X80                ; Mailbox size
00000200  0000   135          MAX_MSG_LEN   = 512                 ; maximum message length
00000001  0000   136          TIME_ID_1     = 1                   ; Timer id to prevent hung
00000002  0000   137          TIME_ID_2     = 2                   ; Timer id to prevent hung
00000003  0000   138          RW_TIME_ID    = 3                   ; Timer to prevent hung when Read/write
00000010  0000   139          LIMIT         = 16                  ; Loop count for each message length
00000008  0000   140          RECV_EFN      = 8
00000005  0000   141          XMIT_EFN      = 5                   ; EFN for QIO write
00000006  0000   142          ATTN_DELV     = 6                   ; EFN for attention AST delivered
00000007  0000   143          MBXAST_DELV   = 7                   ; EFN for mailbox AST delivered
00000040  0000   144          ATTN_DELM     = 1@ATTN_DELV         ; EFN mask for attention ast deliver
00000080  0000   145          MBXAST_DELM   = 1@MBXAST_DELV       ; EFN mask for mailbox ast deliver
00000064  0000   146          PRM           = 100                 ; AST parameter for test
00000000  0000   147          DEVDEP_SIZE   = 0                   ; Size of device dependent part of UETUNT
00000000  0000   148          WRITE_SIZE    = 0                   ; Size of device write buffer
00000000  0000   149          READ_SIZE     = 0                   ; Size of device read buffer
          0000   150
          0000   151          PAGES = <<UETUNT$C_INDSIZ+-         ; Add together all of the pieces...
          0000   152                  DEVDEP_SIZE+-               ; ...which make up a UETP unit block...
          0000   153                  WRITE_SIZE+-                ; ...to give to the $EXPREG service below
          0000   154                  READ_SIZE+-
00000001  0000   155                  511>7512>
          0000   156
0000001B  0000   157          ESC = ^X1B                          ; ESC character
```

```
                                0000   159          .SBTTL  Read-Only Data
                            00000000   160          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000   161
                                0000   162 ACNT_NAME:
53 45 54 53 59 53 00000008'010E0000'    163          .ASCID  /SYSTEST/                  ; Process name on exit
                                  54    000E
                                000F   164
                                000F   165 TEST_NAME:
4D 4F 43 54 45 55 00000017'010E0000'    166          .ASCID  /UETCOMS00/                ; This test name
                              30 30 53  001D
                                0020   167
                                0020   168 SUPDEV_GBLSEC:
50 55 53 54 45 55 00000028'010E0000'    169          .ASCID  /UETSUPDEV/                ; How we access UETSUPDEV.DAT
                              56 45 44  002E
                                0031   170
                                0031   171 CONTROLLER:
41 4E 4C 52 54 43 00000039'010E0000'    172          .ASCID  /CTRLNAME/                 ; Logical name of controller
                                 45 4D  003F
                                0041   173
                                0041   174 MODE:
      45 44 4F 4D 00000049'010E0000'    175          .ASCID  /MODE/                     ; Run mode logical name
                                004D   176
                                004D   177 NO_RMS_AST_TABLE:                          ; List of errors for which...
                        00000000'       178          .LONG   RMS$_BLN                  ; ...RMS cannot deliver an AST...
                        00000000'       179          .LONG   RMS$_BUSY                 ; ...even if one has an ERR= arg
                        00000000'       180          .LONG   RMS$_CDA                  ; Note that we can search table...
                        00000000'       181          .LONG   RMS$_FAB                  ; ...via MATCHC since <31:16>...
                        00000000'       182          .LONG   RMS$_RAB                  ; ...pattern can't be in <15:0>
                        00000014        183 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0061   184
                                0061   185 SYS$INPUT:
4E 49 24 53 59 53 00000069'010E0000'    186          .ASCID  /SYS$INPUT/               ; Name of device from which...
                              54 55 50  006F                                          ; ...the test can be aborted
                                0072   187
                                0072   188 INPUT_ITMLST:                             ; $GETDVI arg list for SYS$INPUT
                          0020 0040     189          .WORD   64,DVI$_DEVNAM            ; We need the equivalence name
                  0000000C'00000014'    190          .LONG   BUFFER,BUFFER_PTR
                          00000000      191          .LONG   0                         ; Terminate the list
                                0082   192
                                0082   193 CS1:                                      ; Device class and type control string
21 20 42 58 32 21 0000008A'010E0000'    194          .ASCID  /!2XB !2XB /
                              20 42 58 32  0090
                                0094   195
                                0094   196 CS3:                                      ; Device class-only control string
2A 20 42 58 32 21 0000009C'010E0000'    197          .ASCID  /!2XB **/
                                 2A     00A2
                                00A3   198
                                00A3   199 CNTRLCMSG:
65 74 72 6F 62 41 000000AB'010E0000'    200          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64  00B1
                  43 2F 4C 52 54 43 20  00BD
                                00C4   201
                                00C4   202 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000CC'010E0000'    203          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 61 72 74  00D2
                  2E 64 65 69 66 69  00DE
                                00E4   204
```

```
                                                00E4   205 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000EC'010E0000'            00E4   206         .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74             00F2
72 61 6D 20 2C 53 41 21 20 72 65 6C             00FE
61 73 75 6E 75 20 73 61 20 64 65 6B             010A
4E 49 54 45 55 20 6E 69 20 65 6C 62             0116
          2E 54 41 44 2E 56 45 44 49            0122
                                                012B   207
                                                012B   208 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000133'010E0000'            012B   209         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74             0139
2E 67 6E 69 74 73 65 74 20 72 6F 66             0145
                                                0151   210
                                                0151   211 ILLEGAL_REC:
61 67 65 6C 6C 49 00000159'010E0000'            0151   212         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C             015F
20 65 6C 69 66 20 6E 69 20 74 61 6D             016B
41 44 2E 56 45 44 49 4E 49 54 45 55             0177
                              21 54             0183
                                                0185   213
                                                0185   214 PASS_MSG:
66 6F 20 64 6E 45 0000018D'010E0000'            0185   215         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20             0193
61 72 65 74 69 20 4C 55 21 20 68 74             019F
44 25 21 20 74 61 20 73 6E 6F 69 74             01AB
                              2E                01B7
                                                01B8   216
                                                01B8   217 INIDEV_UPDERR:                    ; Error during exit handler
20 72 6F 72 72 45 000001C0'010E0000'            01B8   218         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75             01C6
   2E 54 41 44 2E 56 45 44 49 4E 49             01D2
                                                01DD   219
                                                01DD   220 THREEMIN:                         ; 3 minute delta time
                FFFFFFFF 94B62E00               01DD   221         .LONG   -10*1000*1000*180,-1
                                                01E5   222
                                                01E5   223 ONEMIN:                           ; 1 minute delta time
                FFFFFFFF DC3CBA00               01E5   224         .LONG   -10*1000*1000*60,-1
                                                01ED   225
                                                01ED   226 UNIT_DESC:                        ; Descriptor used to convert unit #
                        00000005                01ED   227         .LONG   5
                        0000001A'               01F1   228         .ADDRESS BUFFER+6
                                                01F5   229
                                                01F5   230 CONT_DESC:                        ; Descriptor used to convert controller...
                    0000 0028                   01F5   231         .WORD   REC_SIZE,0        ; ...from lowercase to uppercase
                        00000014'               01F9   232         .ADDRESS BUFFER
                                                01FD   233
                                                01FD   234 FILE:                             ; Fills in RMS_ERR_STRING
      65 6C 69 66 00000205'010E0000'            01FD   235         .ASCID  /file/
                                                0209   236
                                                0209   237 RECORD:                           ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000211'010E0000'            0209   238         .ASCID  /record/
                                                0217   239
                                                0217   240 RMS_ERR_STRING:                   ; Announces an RMS error
41 21 20 53 4D 52 0000021F'010E0000'            0217   241         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53             0225
                  44 41 21 20 65 6C 69          0231
                                                0238   242
```

```
                                                0238   243 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43             0238   244         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65             0244
                                    20          0250
                                00000019        0251   245         PMTSIZ = .-PROMPT
                                                0251   246
                                                0251   247 RECV_ERR_MSG:
76 69 65 63 65 52 00000259'010E0000'            0251   248         .ASCID  /Received message error, good data is !XB, bad data is !XB /
65 20 65 67 61 73 73 65 6D 20 64 65             025F
64 20 64 6F 6F 67 20 2C 72 6F 72 72             026B
20 2C 42 58 21 20 73 69 20 61 74 61             0277
20 73 69 20 61 74 61 64 20 64 61 62             0283
                        20 42 58 21             028F
                                                0293   249
                                                0293   250 ASTPAR_ERRMSG:
20 72 6F 72 72 45 0000029B'010E0000'            0293   251         .ASCID  /Error in passed AST parameter of QIO start or setattn/
53 41 20 64 65 73 73 61 70 20 6E 69             02A1
20 72 65 74 65 6D 61 72 61 70 20 54             02AD
74 72 61 74 73 20 4F 49 51 20 66 6F             02B9
   6E 74 74 61 74 65 73 20 72 6F 20             02C5
                                                02D0   252
                                                02D0   253 MBX_ERRMSG:
20 67 6E 6F 72 57 000002D8'010E0000'            02D0   254         .ASCID  /Wrong message type in the associated mailbox/
65 70 79 74 20 65 67 61 73 73 65 6D             02DE
6F 73 73 61 20 65 68 74 20 6E 69 20             02EA
62 6C 69 61 6D 20 64 65 74 61 69 63             02F6
                              78 6F             0302
                                                0304   255
                                                0304   256 ERR_FATAL_MSG:
65 70 78 65 6E 55 0000030C'010E0000'            0304   257         .ASCID  /Unexpected hardware or software error occurred/
72 61 77 64 72 61 68 20 64 65 74 63             0312
72 61 77 74 66 6F 73 20 72 6F 20 65             031E
75 63 63 6F 20 72 6F 72 72 65 20 65             032A
                        64 65 72 72             0336
                                                033A   258
                                                033A   259 ERR_LOST_MSG:
6C 20 61 74 61 44 00000342'010E0000'            033A   260         .ASCID  /Data lost because message longer than maximum message size/
20 65 73 75 61 63 65 62 20 74 73 6F             0348
67 6E 6F 6C 20 65 67 61 73 73 65 6D             0354
69 78 61 6D 20 6E 61 68 74 20 72 65             0360
20 65 67 61 73 73 65 6D 20 6D 75 6D             036C
                        65 7A 69 73             0378
                                                037C   261
                                                037C   262 ERR_START_MSG:
20 72 6F 72 72 45 00000384'010E0000'            037C   263         .ASCID  /Error because DDCMP START message received/
4D 43 44 44 20 65 73 75 61 63 65 62             038A
73 73 65 6D 20 54 52 41 54 53 20 50             0396
64 65 76 69 65 63 65 72 20 65 67 61             03A2
                                                03AE   264
                                                03AE   265 ERR_MAINT_MSG:
20 72 6F 72 72 45 000003B6'010E0000'            03AE   266         .ASCID  /Error because DDCMP maintenance message received/
4D 43 44 44 20 65 73 75 61 63 65 62             03BC
63 6E 61 6E 65 74 6E 69 61 6D 20 50             03C8
65 72 20 65 67 61 73 73 65 6D 20 65             03D4
                        64 65 76 69 65 63       03E0
                                                03E6   267
                                                03E6   268 STS_ORUN_MSG:
```

```
                                                                                 269          .ASCID  /Data overrun, data received but lack of receive buffer/
6F 20 61 74 61 44 000003EE'010E0000'  03E6
61 74 61 64 20 2C 6E 75 72 72 65 76   03F4
75 62 20 64 65 76 69 65 63 65 72 20   0400
65 72 20 66 6F 20 6B 63 61 6C 20 74   040C
72 65 66 66 75 62 20 65 76 69 65 63   0418
                                      0424   270
                                      0424   271 STS_DCHK_MSG:
63 20 61 74 61 44 0000042C'010E0000'  0424   272          .ASCID  /Data check, retransmission threshold exceeded/
6E 61 72 74 65 72 20 2C 6B 63 65 68   0432
72 68 74 20 6E 6F 69 73 73 69 6D 73   043E
65 65 63 78 65 20 64 6C 6F 68 73 65   044A
                        64 65 64      0456
                                      0459   273
                                      0459   274 STS_TIMO_MSG:
20 50 4D 43 44 44 00000461'010E0000'  0459   275          .ASCID  /DDCMP timeout/
                  74 75 6F 65 6D 69 74 0467
                                      046E   276
                                      046E   277 STS_DISC_MSG:
73 20 61 74 61 44 00000476'010E0000'  046E   278          .ASCID  /Data set ready modem line went from on to off/
64 6F 6D 20 79 64 61 65 72 20 74 65   047C
74 6E 65 77 20 65 6E 69 6C 20 6D 65   0488
20 6F 74 20 6E 6F 20 6D 6F 72 66 20   0494
                        66 66 6F      04A0
                                      04A3   279
                                      04A3   280 NO_WAIT_READ:
67 61 73 73 65 4D 000004AB'010E0000'  04A3   281          .ASCID  /Message available but no waiting read request/
20 65 6C 62 61 6C 69 61 76 61 20 65   04B1
69 74 69 61 77 20 6F 6E 20 74 75 62   04BD
75 71 65 72 20 64 61 65 72 20 67 6E   04C9
                        74 73 65      04D5
                                      04D8   282
                                      04D8   283 ERR_ATTN_MSG:
74 6E 65 74 74 41 000004E0'010E0000'  04D8   284          .ASCID  /Attention AST delivered for unknown reasons/
69 6C 65 64 20 54 53 41 20 6E 6F 69   04E6
6E 75 20 72 6F 66 20 64 65 72 65 76   04F2
6E 6F 73 61 65 72 20 6E 77 6F 6E 6B   04FE
                              73      050A
                                      050B   285
                                      050B   286 ATTN_MBX_MSG:
      2E 53 41 21 00000513'010E0000'  050B   287          .ASCID  \!AS.\-
74 61 69 63 6F 73 73 41 5F 21 2F 21   0517   288                  \!/!_Associated mailbox has type=MSG$_XM_!AC on !AC, unit !UW.\
68 20 78 6F 62 6C 69 61 6D 20 64 65   0523
24 47 53 4D 3D 65 70 79 74 20 73 61   052F
21 20 6E 6F 20 43 41 21 5F 4D 58 5F   053B
57 55 21 20 74 69 6E 75 20 2C 43 41   0547
                              2E      0553
                                      0554   289
                                      0554   290 ATTN_MBX_TYPES:
                                000B  0554   291          .WORD   MSG$_XM_DATAVL
                                000C  0556   292          .WORD   MSG$_XM_SHUTDN
                                000D  0558   293          .WORD   MSG$_XM_ATTN
                                000B  055A   294          .WORD   MSG$_XM_DATAVL          ; Allows MATCHC to distinguish...
                                      055C   295                                          ; ...between last entry and unknown
                           00000008   055C   296 ATTN_MBX_TYPES_LENGTH = .-ATTN_MBX_TYPES
                                      055C   297
                                      055C   298 ATTN_MBX_TYPES_NAMES:
                           00000583'  055C   299          .ADDRESS ATTN_MBX_TYPES_UNKNOWN ; Duplicate entry here...
```

```
                00000583' 0560    300          .ADDRESS ATTN_MBX_TYPES_UNKNOWN ; ...makes later coding easier
                0000057E' 0564    301          .ADDRESS ATTN_MBX_TYPES_ATTN
                00000577' 0568    302          .ADDRESS ATTN_MBX_TYPES_SHUTDN
                00000570' 056C    303          .ADDRESS ATTN_MBX_TYPES_DATAVL
                          0570    304
                          0570    305 ATTN_MBX_TYPES_DATAVL:
      4C 56 41 54 41 44 00' 0570    306          .ASCIC  /DATAVL/
                       06   0570
                          0577    307 ATTN_MBX_TYPES_SHUTDN:
      4E 44 54 55 48 53 00' 0577    308          .ASCIC  /SHUTDN/
                       06   0577
                          057E    309 ATTN_MBX_TYPES_ATTN:
            4E 54 54 41 00' 057E    310          .ASCIC  /ATTN/
                       04   057E
                          0583    311 ATTN_MBX_TYPES_UNKNOWN:
   6E 77 6F 6E 6B 6E 75 00' 0583    312          .ASCIC  /unknown/
                       07   0583
```

```
                                0588   314          .SBTTL  Read/Write Data
                         00000000      315          .PSECT  RWDATA,WRT,NOEXE,PAGE
                                0000   316
                                0000   317 TTCHAN:                                ; Channel associated with ctrl. term.
                           0000 0000   318          .WORD   0
                                0002   319
                                0002   320 FLAG:                                  ; Miscellaneous flag bits
                           0000 0002   321          .WORD   0                     ; (See Equated Symbols for definitions)
                                0004   322
                                0004   323 FAO_BUF:                               ; FAO output string descriptor
                      0000 0084 0004   324          .WORD   TEXT_BUFFER,0
                       00000014' 0008  325          .ADDRESS BUFFER
                                000C   326
                                000C   327 BUFFER_PTR:                           ; Fake .ASCID buffer for misc. strings
                      0000 0084 000C   328          .WORD   TEXT_BUFFER,0         ; A word for length, a word for desc.
                       00000014' 0010  329          .ADDRESS BUFFER
                                0014   330
                                0014   331 BUFFER:                               ; FAO output and other misc. buffer
                         00000098 0014  332          .BLKB   TEXT_BUFFER
                                0098   333
                                0098   334 DEVDSC:                               ; Device name descriptor
                      0000 000A 0098   335          .WORD   MAX_DEV_DESIG,0
                       000000B7' 009C  336          .ADDRESS DEV_NAME
                                00A0   337
                                00A0   338 PROCESS_NAME:                        ; Process name
53 4D 4F 43 000000A8'010E0000' 00A0   339          .ASCID  /COMS/
                         0000000B 00AC  340          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                         000000B7 00AC  341          .BLKB   PROCESS_NAME_FREE
                                00B7   342
                                00B7   343 DEV_NAME:                            ; Device name buffer
                         000000C6 00B7  344          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                         0000000F 00C6  345          NAME_LEN = .-DEV_NAME
                                00C6   346
                                00C6   347 DIB:                                 ; Device Information Block
                      0000 0074 00C6   348          .WORD   DIB$K_LENGTH,0
                       000000CE' 00CA  349          .ADDRESS DIBBUF
                                00CE   350 DIBBUF:
                         00000142 00CE  351          .BLKB   DIB$K_LENGTH
                                0142   352
                                0142   353 ERROR_COUNT:                          ; Cumulative error count at runtime
                         00000000 0142  354          .LONG   0
                                0146   355
                                0146   356 STATUS:                              ; Status value on program exit
                         00000000 0146  357          .LONG   0
                                014A   358
                                014A   359 QUAD_STATUS:                         ; IO status block for misc sys. svcs.
              00000000 00000000 014A   360          .QUAD   0
                                0152   361
                                0152   362 INADDRESS:                           ; $CRMPSC address storage
              00000000 00000000 0152   363          .LONG   0,0
                                015A   364
                                015A   365 OUTADDRESS:
              00000000 00000000 015A   366          .LONG   0,0
                                0162   367
                                0162   368 UNIT_NUMBER:                         ; Current dev unit number
                           0000 0162   369          .WORD   0
                                0164   370
```

```
                     0164   371 DEVNAM_LEN:                             ; Current device name length
          0000       0164   372         .WORD   0
                     0166   373
                     0166   374 ITERATION:                              ; # of times all tests were executed
      00000000       0166   375         .LONG   0
                     016A   376
                     016A   377 PASS:                                   ; Pass count
      00000000       016A   378         .LONG   0
                     016E   379
                     016E   380 MSG_BLOCK:                              ; Auxiliary $GETMSG info
      00000172       016E   381         .BLKB   4
                     0172   382
                     0172   383 EXIT_DESC:                             ; Exit handler descriptor
      00000000       0172   384         .LONG   0
      00000C17'      0176   385         .ADDRESS EXIT_HANDLER
      00000001       017A   386         .LONG   1
      00000146'      017E   387         .ADDRESS STATUS
                     0182   388
                     0182   389 ARG_COUNT:                             ; Argument counter used by ERROR_EXIT
      00000000       0182   390         .LONG   0
                     0186   391
                     0186   392 MBXCHAN:                               ; Associated mailbox channel
          0000       0186   393         .WORD   0
                     0188   394
                     0188   395 XMMBX_DESC:                            ; Mailbox logical name descriptor
      00000007'      0188   396         .LONG   MBX_LOGNAMSIZ
      00000190'      018C   397         .LONG   MBXLOGNAM
                     0190   398
                     0190   399 MBXLOGNAM:                            ; Mailbox logical name
  58 42 4D 5F 43 4D 44 0190   400         .ASCII  /DMC_MBX/
                     0197   401
            00000007 0197   402 MBX_LOGNAMSIZ = .-MBXLOGNAM
                     0197   403
                     0197   404 XM_CHAN:                              ; DMC/R channel
              0000   0197   405         .WORD   0
                     0199   406
                     0199   407 DEVCHAR_BLK:                          ; Device char block
  00000000 00000000  0199   408         .QUAD   0
                     01A1   409
                     01A1   410 EF_MASK:                              ; Mask for EFN wait
      00000000       01A1   411         .LONG   0
                     01A5   412
                     01A5   413 XM_IOSB:                              ; QIO IO status block
      000001AD       01A5   414         .BLKQ   1
                     01AD   415
                     01AD   416 RECV_IOSB:                            ; QIO read message IO status block
      000001B5       01AD   417         .BLKQ   1
                     01B5   418
                     01B5   419 XMIT_BUF:                             ; Transmit buffer
      000003B5       01B5   420         .BLKB   MAX_MSG_LEN
                     03B5   421
                     03B5   422 RECV_BUF:                             ; Receive buffer
      000005B5       03B5   423         .BLKB   MAX_MSG_LEN
                     05B5   424
                     05B5   425 MBX_BUF:                              ; mailbox fuffer
      00000635       05B5   426         .BLKB   MBXSIZE
                     0635   427
```

```
                     0635   428 BAD_DATA:                                   ; Received wrong data
                00   0635   429         .BYTE   0
                     0636   430
                     0636   431 GOOD_DATA:                                  ; Data sent (good)
                00   0636   432         .BYTE   0
                     0637   433
                     0637   434
                     0637   435 ;
                     0637   436 ; Head of self-relative UETP unit block queue.
                     0637   437 ;
                     0637   438         .ALIGN QUAD
                     0638   439
                     0638   440 UNIT_LIST:                                  ; Head of unit block circular list
00000000 00000000    0638   441         .QUAD   0
                     0640   442
                     0640   443 NEW_NODE:                                   ; Newly acquired node address
00000000 00000000    0640   444         .QUAD   0
```

```
                     0648   446          .SBTTL  RMS-32 Data Structures
                     0648   447          .ALIGN  LONG
                     0648   448
                     0648   449 SYSIN_FAB:                                  ; Allocate FAB for SYS$INPUT
                     0648   450          $FAB-
                     0648   451          FNM = <SYS$INPUT>
                     0698   452
                     0698   453 SYSIN_RAB:                                  ; Allocate RAB for SYS$INPUT
                     0698   454          $RAB-
                     0698   455          FAB = SYSIN_FAB,-
                     0698   456          ROP = PMT,-
                     0698   457          PBF = PROMPT,-
                     0698   458          PSZ = PMTSIZ,-
                     0698   459          UBF = DEV_NAME,-
                     0698   460          USZ = NAME_LEN
                     06DC   461
                     06DC   462 INI_FAB:                                    ; Allocate FAB for UETINIDEV
                     06DC   463          $FAB-
                     06DC   464          FAC = <GET,PUT,UPD>,-
                     06DC   465          RAT = CR,-
                     06DC   466          SHR = <GET,PUT,UPI>,-
                     06DC   467          FNM = <UETINIDEV.DAT>
                     072C   468
                     072C   469 INI_RAB:                                    ; Allocate RAB for UETINIDEV
                     072C   470          $RAB-
                     072C   471          FAB = INI_FAB,-
                     072C   472          RBF = BUFFER,-
                     072C   473          UBF = BUFFER,-
                     072C   474          USZ = REC_SIZE
                     0770   475
                     0770   476 DDB_RFA:                                    ; RFA storage for INI_RAB
00000776             0770   477          .BLKB   6
                     0776   478
                     0776   479          .ALIGN  LONG
                     0778   480 SUP_FAB:                                    ; Allocate FAB for UETSUPDEV
                     0778   481          $FAB-
                     0778   482          FAC = GET,-
                     0778   483          SHR = <UPI,GET>,-
                     0778   484          RAT = CR,-
                     0778   485          FOP = UFO,-
                     0778   486          FNM = <UETSUPDEV.DAT>
                     07C8   487
                     07C8   488 ;
                     07C8   489 ; Dummy FAB and RAB to copy to the UETP unit blocks
                     07C8   490 ; The following FAB and RAB must be contiguous and in this order!
                     07C8   491 ;
                     07C8   492
                     07C8   493 DUMMY_FAB:
                     07C8   494          $FAB
                     0818   495
                     0818   496 DUMMY_RAB:
                     0818   497          $RAB    RSZ = WRITE_SIZE,-
                     0818   498                  USZ = READ_SIZE
```

```
                        085C   500          .SBTTL  Main Program
                    00000000   501          .PSECT  COMS,EXE,NOWRT,PAGE
                        0000   502
                        0000   503          .DEFAULT DISPLACEMENT,WORD
                        0000   504 ;+
                        0000   505 ;        Start up the DMC/DMR test.
                        0000   506 ;-
                        0000   507
                        0000   508 .ENTRY UETCOMS00,^M<>                        ; Entry mask
                        0002   509
            6D  09C0'CF  DE     0002   510          MOVAL   SSERROR,(FP)         ; Declare exception handler
                                0007   511          $SETSFM_S ENBFLG = #1        ; Enable system service failure mode
                                0010   512          $DCLEXH_S DESBLK = EXIT_DESC ; Declare an exit handler
                                001B   513
                                001B   514          $OPEN   FAB = SYSIN_FAB,-    ; Open SYS$INPUT
                                001B   515                  ERR = RMS_ERROR
                                002A   516          $CONNECT RAB = SYSIN_RAB,-   ; Connect RAB to SYS$INPUT
                                002A   517                   ERR = RMS_ERROR
                    02          E1     0039   518          BBC     S^#DEV$V_TRM,- ; BR if SYS$INPUT is NOT a terminal
                1E 0688'CF         003B   519                  SYSIN_FAB+FAB$L_DEV,10$
                                003F   520          $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                                003F   521                    RSLLEN = DEVNAM_LEN,- ; ...a logial name...
                                003F   522                    RSLBUF = DEVDSC       ; ...for the controller to test
                    01      50  D1     0058   523          CMPL    R0,#SS$_NORMAL ; Was a controller specified?
                            2E  13     005B   524          BEQL    PROC_CONT_NAME ; BR if it was - go process it
                                005D   525 10$:
                                005D   526          $GET    RAB = SYSIN_RAB,-    ; Read SYS$INPUT...
                                005D   527                  ERR = RMS_ERROR      ; ...for the controller name
                06BA'CF         B0     006C   528          MOVW    SYSIN_RAB+RAB$W_RSZ,- ; Save the name length
                0164'CF         0070   529                  DEVNAM_LEN
                            16  12     0073   530          BNEQ    PROC_CONT_NAME ; BR if we got something
            0146'CF         14  D0     0075   531          MOVL    #SS$_BADPARAM,STATUS ; Save an exit status if not
                00C4'CF         DF     007A   532          PUSHAL  NO_CTRLNAME    ; Prepare for message...
                            01  DD     007E   533          PUSHL   #1             ; ...arg count
             00741132       8F  DD     0080   534          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name
                            03  DD     0086   535          PUSHL   #3             ; ...arg count
                        0AE5    31     0088   536          BRW     ERROR_EXIT     ; ...go tell of bad setup
                                008B   537
                                008B   538 PROC_CONT_NAME:
        0098'CF     0164'CF     3C     008B   539          MOVZWL  DEVNAM_LEN,DEVDSC ; Set the device name length
                    0098'CF     DF     0092   540          PUSHAL  DEVDSC         ; Make sure...
                    0098'CF     DF     0096   541          PUSHAL  DEVDSC         ; ...that the specified controller...
          00000000'GF     02     FB     009A   542          CALLS   #2,G^STR$UPCASE ; ...is all uppercase for later comaparison
        52  0098'CF     01     C1     00A1   543          ADDL3   #1,DEVDSC,R2   ; Estimate the eventual...
            00A0'CF     52     A0     00A7   544          ADDW2   R2,PROCESS_NAME ; ...process name length (incl. "_")
                               DE     00AC   545          MOVAL   PROCESS_NAME+8- ; Locate first available byte...
                               00AD   546                  +MAX_PROC_NAME-      ; ...in process name handle...
                50  00AC'CF        00AD   547                  -PROCESS_NAME_FREE,R0 ; ...for device name
                                0B  C3     00B1   548          SUBL3   #PROCESS_NAME_FREE,- ; Will the device name fit...
                            51  52     00B3   549                  R2,R1          ; ...in the remaining space?
                                08  15     00B5   550          BLEQ    10$        ; BR if it will
                            50  51  C2     00B7   551          SUBL2   R1,R0      ; Overwrite handle otherwise...
            00A0'CF         0F  B0     00BA   552          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                                00BF   553 10$:
                      80    5F  8F     00BF   554          MOVB    #^A/_/,(R0)+   ; Separate handle from device name
        60  00B7'CF     0098'CF  28    00C3   555          MOVC3   DEVDSC,DEV_NAME,(R0) ; Concatenate handle with device name
                                7E     D4     00CB   556          CLRL    -(SP)          ; Set the time stamp flag
```

```
                000F'CF    DF   00CD    557          PUSHAL  TEST_NAME                       ; Set the test name
                   02      DD   00D1    558          PUSHL   #2                              ; Push the argument count
          00741039 8F      DD   00D3    559          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
00000000'GF       04      FB   00D9    560          CALLS   #4,G^LIB$SIGNAL                 ; Print the startup message
     0002'CF      08      A8   00E0    561          BISW2   #BEGIN_MSGM,FLAG                ; Set flag so we don't print it again
                          00E5    562          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETCOMS00_x
                          00F0    563
                   02     E1   00F0    564          BBC     S^#DEV$V_TRM,-                  ; BR if SYS$INPUT is NOT a terminal
                66 0688'CF         00F2    565                  SYSIN_FAB+FAB$L_DEV,20$
                          00F6    566          $GETDVI_S DEVNAM = SYS$INPUT,-          ; Get the name of...
                          00F6    567                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                          00F6    568                    ITMLST = INPUT_ITMLST,-
                          00F6    569                    IOSB   = QUAD_STATUS
                45 014A'CF    E9   0112    570          BLBC    QUAD_STATUS,20$                 ; Avoid CTRL/C handler if any error
                          0117    571          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                          0117    572                    CHAN   = TTCHAN
                          0128    573          $QIOW_S CHAN     = TTCHAN,-             ; Enable CTRL/C AST's...
                          0128    574                  FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                          0128    575                  P1       = CCASTHAND
                   00A0'CF    DF   0149    576          PUSHAL  PROCESS_NAME                    ; ...and tell the user...
                      01      DD   014D    577          PUSHL   #1                              ; ...
          0074832B 8F      DD   014F    578          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
00000000'GF       03      FB   0155    579          CALLS   #3,G^LIB$SIGNAL                 ; ...
                          015C    580 20$:
```

```
                                015C    582 ;
                                015C    583 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                015C    584 ; and unit configuration and lets us know if the setup to run this test was
                                015C    585 ; done correctly.
                                015C    586 ;
                                015C    587         $OPEN   FAB = INI_FAB,-             ; Open file "UETINIDEV.DAT"
                                015C    588                 ERR = RMS_ERROR
                                016B    589         $CONNECT RAB = INI_RAB,-            ; Connect the RAB and FAB
                                016B    590                  ERR = RMS_ERROR
                                017A    591         $MGBLSC_S  INADR = INADDRESS,-      ; Connect to UETSUPDEV global section
                                017A    592                    RETADR = OUTADDRESS,-
                                017A    593                    GSDNAM = SUPDEV_GBLSEC,-
                                017A    594                     FLAGS = #SEC$M_EXPREG
        00000978 8F    50    D1 0199    595         CMPL    R0,#SS$_NOSUCHSEC           ; Was the section already there?
                       37    12 C1A0    596         BNEQ    30$                         ; BR if it was...
                             01A2    597         $OPEN   FAB = SUP_FAB,-             ; ...else open "UETSUPDEV.DAT"
                             01A2    598                 ERR = RMS_ERROR
                             01B1    599         $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                             01B1    600                   INADR = INADDRESS,-
                             01B1    601                 RETADR = OUTADDRESS,-
                             01B1    602                 GSDNAM = SUPDEV_GBLSEC,-
                             01B1    603                  FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                             01D9    604 30$:
     56 015E'CF    015A'CF    C3 01D9    605         SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                             01E1    606
                             01E1    607 FIND_IT:
                             01E1    608         $GET    RAB = INI_RAB,-             ; Get the first record
                             01E1    609                 ERR = RMS_ERROR
                01F5'CF      DF 01F0    610         PUSHAL  CONT_DESC                   ; Make sure...
                01F5'CF      DF 01F4    611         PUSHAL  CONT_DESC                   ; ...that the controller name...
          00000000'GF        02 01F8    612         CALLS   #2,G^STR$UPCASE             ; ...is all uppercase letters
           0014'CF    44 8F    91 01FF    613         CMPB    #^A/D/,BUFFER               ; Is this a DDB?
                       27    13 0205    614         BEQL    10$                         ; Go on if not
           0014'CF    45 8F    91 0207    615         CMPB    #^A/E/,BUFFER               ; Is this the end of the file?
                       D2    12 020D    616         BNEQ    FIND_IT                     ; Continue on if not
                0098'CF      DF 020F    617         PUSHAL  DEVDSC                      ; Push device not supported message
                00A0'CF      DF 0213    618         PUSHAL  PROCESS_NAME                ; Parameters on the stack
                             02 DD 0217    619         PUSHL   #2
                  00748333 8F    DD 0219    620         PUSHL   #UETP$_DENOSU
                             02    F0 021F    621         INSV    #STS$K_ERROR,-              ; Set the severity code...
                             00       0221    622                 #STS$V_SEVERITY,-
                          6E    03       0222    623                 #STS$S_SEVERITY,(SP)
                0146'CF      6E    D0 0224    624         MOVL    (SP),STATUS                 ; ...and save it as the exit status
                             04    DD 0229    625         PUSHL   #4
                           0942    31 022B    626         BRW     ERROR_EXIT                  ; Exit in error
                                   022E    627 10$:
00B7'CF    001A'CF    0164'CF    29 022E    628         CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME    ; Is this the right controller?
                             A7    12 0238    629         BNEQ    FIND_IT                     ; BR if not
     0770'CF    073C'CF       06    28 023A    630         MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
           0018'CF    54 8F    91 0242    631         CMPB    #^A/T/,BUFFER+4             ; Can we test this controller?
                       2F    13 0248    632         BEQL    FOUND_IT                    ; BR if we can...
                             024A    633         $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                             024A    634                 OUTLEN = BUFFER_PTR,-
                             024A    635                 OUTBUF = FAO_BUF,-
                             024A    636                 P1     = #DEVDSC
                0146'CF      14 D0 0263    637         MOVL    #SS$_BADPARAM,STATUS        ; Set return status
                000C'CF      DF 0268    638         PUSHAL  BUFFER_PTR                  ; ...
```

```
                                                M 9
                     01   DD   026C   639          PUSHL   #1                              ; ...
           00741132  8F   DD   026E   640          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...
                     03   DD   0274   641          PUSHL   #3                              ; ...
                   08F7   31   0276   642          BRW     ERROR_EXIT                      ; We can't test what we can't test
                                0279   643
                                0279   644 FOUND_IT:
                                0279   645          $GET    RAB = INI_RAB,-                 ; Get a record
                                0279   646                  ERR = RMS_ERROR
                01F5'CF    DF   0288   647          PUSHAL  CONT_DESC                       ; Make sure...
                01F5'CF    DF   028C   648          PUSHAL  CONT_DESC                       ; ...that this line...
      0000C000'GF    02   FB   0290   649          CALLS   #2,G^STR$UPCASE                 ; ...is all uppercase letters
       0014'CF  55 8F   91   0297   650          CMPB    #^A/U/,BUFFER                   ; Is this a UCB?
                   24   13   029D   651          BEQL    30$                             ; BR if it is
       0014'CF  44 8F   91   029F   652          CMPB    #^A/D/,BUFFER                   ; Is this a DDB?
                   19   13   02A5   653          BEQL    20$                             ; BR if yes
       0014'CF  45 8F   91   02A7   654          CMPB    #^A/E/,BUFFER                   ; Is this the end?
                   11   13   02AD   655          BEQL    20$                             ; BR if yes
                                02AF   656 10$:
                0151'CF    DF   02AF   657          PUSHAL  ILLEGAL_REC                     ; Then this is an error in the record
                     01    DD   02B3   658          PUSHL   #1                              ; Push the error message
           00741132  8F    DD   02B5   659          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Push the signal name
                     03    DD   02BB   660          PUSHL   #3                              ; Push the temp arg count
                   08B0    31   02BD   661          BRW     ERROR_EXIT                      ; Finish for good
                                02C0   662 20$:
                   0126    31   02C0   663          BRW     ALL_SET                         ; Found DDB or END
                                02C3   664 30$:
       0018'CF  54 8F   91   02C3   665          CMPB    #^A/T/,BUFFER+4                 ; Is the unit testable?
                   AE   12   02C9   666          BNEQ    FOUND_IT                        ; BR if not
                     01    DD   02CB   667          PUSHL   #1                              ; Flag to ignore blanks when converting
                     02    DD   02CD   668          PUSHL   #2                              ; Set byte size of results
                0162'CF    DF   02CF   669          PUSHAL  UNIT_NUMBER                     ; Set address to receive word
                01ED'CF    DF   02D3   670          PUSHAL  UNIT_DESC                       ; Push string address
          00000000'GF    04   FB   02D7   671          CALLS   #4,G^OTS$CVT_TI_L               ; Convert ASCII unit # to decimal
                     CE   50   E9   02DE   672          BLBC    R0,10$                          ; Don't allow bogus unit to pass
                05   20   3B   02E1   673          SKPC    #^A/ /,#MAX_UNIT_DESIG,-        ; Find out where unit number really is
                001A'CF         02E4   674                  BUFFER+6
                     50   D7   02E7   675          DECL    R0                              ; Units must all be at least one digit
             61      50   30   3B   02E9   676          SKPC    #^A/0/,R0,(R1)                  ; Skip leading zeroes on the unit
                     50   D6   02ED   677          INCL    R0                              ; Compensate for DECL above
       0098'CF  0164'CF   50   A1   02EF   678          ADDW3   R0,DEVNAM_LEN,DEVDSC            ; Calculate device'unit string length
             52  0164'CF   3C   02F7   679          MOVZWL  DEVNAM_LEN,R2                   ; Offset to unit number in DEVDSC
          00B7'C2   61      50   28   02FC   680          MOVC3   R0,(R1),DEV_NAME(R2)            ; Append unit number to device
                                0302   681
                   03A8    30   0302   682          BSBW    START_DEV                       ; Assign channel and start the device
                                0305   683
                                0305   684          $GETDEV_S DEVNAM = DEVDSC,-             ; Get the device characteristics
                                0305   685                    PRIBUF = DIB
             57  00D2'CF   9A   031A   686          MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7        ; Save the device class
             58  00D3'CF   9A   031F   687          MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8         ; Save the device type
                                0324   688          $FAO_S  CTRSTR = CS1,-
                                0324   689                  OUTBUF = FAO_BUF,-
                                0324   690                  P1     = R7,-
                                0324   691                  P2     = R8                     ; Make it into a string
    015A'DF  56  0014'CF   06   39   0339   692          MATCHC  #6,BUFFER,R6,@OUTADDRESS        ; Find the device class and type
                     1E   13   0342   693          BEQL    40$                             ; BR if it was found
                                0344   694          $FAO_S  CTRSTR = CS3,-                  ; Try for full class support
                                0344   695                  OUTBUF = FAO_BUF,-
```

```
                                0344   696                     P1 = R7
015A'DF   56   0014'CF   06     39 0357   697          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                         0D     12 0360   698          BNEQ    50$                     ; BR if not found
                                   0362   699 40$:
          55   000F'CF   9A       0362   700          MOVZBL  TEST_NAME,R5            ; Get the test name length
    0017'CF    63   55     29     0367   701          CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                    1F     13     036D   702          BEQL    60$                     ; Br if yes
                                   036F   703 50$:
               0098'CF     DF     036F   704          PUSHAL  DEVDSC                  ; Push device not supported message
               00A0'CF     DF     0373   705          PUSHAL  PROCESS_NAME            ; Parameters on the stack
                    02     DD     0377   706          PUSHL   #2                      ; Push the argument count
            00748333    8F     DD 0379   707          PUSHL   #UETP$_DENOSU
                    02     F0     037F   708          INSV    #STS$K_ERROR,-
                    00            0381   709                  #STS$V_SEVERITY,-
                6E  03            0382   710                  #STS$S_SEVERITY,(SP)    ; Set the severity code...
          0146'CF   6E     D0     0384   711          MOVL    (SP),STATUS             ; ...and save it as the exit status
                    04     DD     0389   712          PUSHL   #4                      ; Push the partial arg count...
                 07E2      31     038B   713          BRW     ERROR_EXIT              ; ...and split this scene
```

```
                     ;+
                     ; The following code dynamically allocates enough memory for a unit block,
                     ; a device dependent parameter area and I/O buffers. The unit block is inserted
                     ; into the queue header UNIT_LIST.  It then initializes the unit block.
                     ; A comment indicates where the device dependent parameters should be
                     ; initialized.  The unit block format is as follows:
                     ;
                     ;                                +---------------+ -------
                     ;       UETUNT$L_FLINK           !               !       ^
                     ;                                +---------------+       !
                     ;       UETUNT$L_BLINK           !               !       !
                     ;                                +-----------+---+       !
                     ;       UETUNT$B_TYPE                        !   !       !
                     ;                                    +-------+---+       !
                     ;       UETUNT$W_SIZE                !   !   !    contains DEVDEP_SIZE + UETUNT$C_INDSIZ
                     ;                                +---+-------+           !
                     ;       UETUNT$B_FLAGS           !   !                   !
                     ;                                +---+   +-------+       !
                     ;       UETUNT$W_CHAN                    !   !   !       !
                     ;                                +-------+-------+       !
                     ;       UETUNT$W_FUNC            !   !   !               !
                     ;                                +-------+-------+       +----- UETUNT$C_SIZE
                     ;       UETUNT$L_ITER            !               !       !
                     ;                                +---------------+       !
                     ;       UETUNT$T_FILSPC          !               !       !
                     ;                                !/\/\/\/\/\/\/\/!       !
                     ;                                NAM$C_MAXRSS bytes      !
                     ;                                !/\/\/\/\/\/\/\/!       !
                     ;                                !               !       !
                     ;       UETUNT$K_FAB             !---------------!       !
                     ;                                !               !       !
                     ;                                !/\/\/\/\/\/\/\/!       !
                     ;                                 FAB$C_BLN bytes        !
                     ;                                !/\/\/\/\/\/\/\/!       !
                     ;                                !               !       !
                     ;       UETUNT$K_RAB             +---------------+       !
                     ;                                !               !       !
                     ;                                !\/\/\/\/\/\/\/\!       !
                     ;                                 RAB$C_BLN bytes        !
                     ;                                !\/\/\/\/\/\/\/\!       !
                     ;                                !               !       v
                     ;       UETUNT$K_DEVDEP          +---------------+ -------
                     ;                                !               !       ^
                     ;                                !\/\/\/\/\/\/\/\!       !
                     ;                                  user defined          +----- DEVDEP_SIZE
                     ;                                !\/\/\/\/\/\/\/\!       !
                     ;                                !               !       v
                     ;                                +---------------+ -------
                     ;    READ/WRITE buffers          !               !       ^
                     ;                                !\/\/\/\/\/\/\/\!       !
                     ;                                  user defined          +----- WRITE_SIZE and READ_SIZE
                     ;                                !/\/\/\/\/\/\/\/!       !
                     ;                                !               !       v
                     ;                                +---------------+ -------
                     ;-
```

```
                                038E    771 60$:
                                038E    772             $EXPREG_S PAGCNT = #PAGES,-     ; Get a new node of demand zero memory
                                038E    773                       RETADR = NEW_NODE
0638'CF    0640'DF    5D        039F    774             INSQTI  @NEW_NODE,UNIT_LIST     ; Put the new node in the unit list
      56   0640'CF    D0        03A6    775             MOVL    NEW_NODE,R6             ; Save a copy of its address
        08 A6    01    90        03AB    776             MOVB    #1,UETUNT$B_TYPE(R6)    ; Set the structure type
           01A4 8F    B0        03AF    777             MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
             09 A6              03B3    778                     UETUNT$W_SIZE(R6)       ; Set the structure size
  14 A6    0098'CF    90        03B5    779             MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
009C'DF    0098'CF    28        03BB    780             MOVC3   DEVDSC,@DEVDSC+4,-
             15 A6              03C2    781                     UETUNT$T_FILSPC+1(R6)   ; Save the device name
           0094 8F    28        03C4    782             MOVC3   #FAB$C_BLN+RAB$C_BLN,-
0110 C6    07C8'CF            03C8    783                     DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
      57   0110 C6    DE        03CE    784             MOVAL   UETUNT$K_FAB(R6),R7     ; Save the FAB address
      58   0160 C6    DE        03D3    785             MOVAL   UETUNT$K_RAB(R6),R8     ; Save the RAB address
        3C A8    57    D0        03D8    786             MOVL    R7,RAB$L_FAB(R8)        ; Set the FAB address in the RAB
             14 A6    90        03DC    787             MOVB    UETUNT$T_FILSPC(R6),-
             34 A7              03DF    788                     FAB$B_FNS(R7)           ; Set the FNS field in the FAB
             15 A6    DE        03E1    789             MOVAL   UETUNT$T_FILSPC+1(R6),-
             2C A7              03E4    790                     FAB$L_FNA(R7)           ; Set the FNA field in the FAB
                                03E6    791 ;
                                03E6    792 ; Set the device dependent parameters in here
                                03E6    793 ;
               FE90    31       03E6    794             BRW     FOUND_IT                ; Do the next UCB
```

```
                     03E9    796 ;
                     03E9    797 ; Arrive here when we have the device configuration.  In normal or loop forever
                     03E9    798 ; mode, set a timer far enough in the future such that we can do a reasonable
                     03E9    799 ; set of tests before the timer expires, but if our device gets hung, the
                     03E9    800 ; program won't waste too much time before noticing.  Let one-shot mode be a
                     03E9    801 ; special case.
                     03E9    802 ;
                     03E9    803 ALL_SET:
      0638'CF    D5  03E9    804          TSTL    UNIT_LIST                       ; Anything to test?
            16   12  03ED    805          BNEQ    10$                             ; BR if yes
      012B'CF    DF  03EF    806          PUSHAL  NOUNIT_SELECTED                 ; Else set up the error message...
            01   DD  03F3    807          PUSHL   #1                              ; ...argument count...
  00741132 8F    DD  03F5    808          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...signal name...
            03   DD  03FB    809          PUSHL   #3                              ; ...and parameter count
      0146'CF    14  03FD    810          MOVL    #SS$_BADPARAM,STATUS            ; Set return status
          076B   31  0402    811          BRW     ERROR_EXIT                      ; ...and give up, complaining
                     0405    812 10$:
      0002'CF    04  A8  0405    813          BISW2   #SAFE_TO_UPDM,FLAG          ; OK safe to update UETINIDEV.DAT now
```

```
                        040A   815          .SBTTL  Test the DMC/DMR
                        040A   816
                        040A   817 START_TEST:
                        040A   818          $QIOW_S -                          ; Enable attention AST
                        040A   819                  CHAN = XM_CHAN,-
                        040A   820                  FUNC = #IO$_SETMODE!IO$M_ATTNAST,-
                        040A   821                  IOSB = XM_IOSB,-
                        040A   822                  ASTADR = CHK_QIO_AST,-
                        040A   823                  ASTPRM = #PRM,-
                        040A   824                  P1 = XM_ATTN_AST
                        0435   825
                        0435   826          $TRNLOG_S LOGNAM = MODE,-          ; Get the run mode
                        0435   827                    RSLLEN = BUFFER_PTR,-
                        0435   828                    RSLBUF = FAO_BUF
                        044E   829
    0014'CF    20   8A  044E   830          BICB2   #LC_BITM,BUFFER            ; Convert to upper case
0014'CF    4F   8F   91  0453   831          CMPB    #^A70/,BUFFER              ; Is this a one shot?
                  0D   12  0459   832          BNEQ    10$                        ; BR if not
    0002'CF    02   A8  045B   833          BISW2   #TEST_OVERM,FLAG           ; End after one iteration
    0002'CF    10   A8  0460   834          BISW2   #MODE_IS_ONEM,FLAG         ; Set mode is "ONE" flag
              0013   31  0465   835          BRW     XMIT_RECV                  ; Skip the 3 min timer
                        0468   836 10$:
                        0468   837          $SETIMR_S DAYTIM = THREEMIN,-       ; Set timer AST to 3 minutes
                        0468   838                    ASTADR = TIME_SUC_OUT    ; The test will do xmit/recv for about
                        047B   839                                             ; 3 minutes
                        047B   840 XMIT_RECV:
      52    AA   8F   9A  047B   841          MOVZBL  #^XAA,R2                   ; Random number 1
      53    2E   9A  047F   842          MOVZBL  #^X2E,R3                   ; Random number 2
57    00000200   8F   D0  0482   843          MOVL    #MAX_MSG_LEN,R7            ; Maximum message length
                        0489   844 10$:
56    01B5'CF    DE  0489   845          MOVAL   XMIT_BUF,R6                ; Transmit buffer address
      54    57   D0  048E   846          MOVL    R7,R4                      ; Message length in bytes
                        0491   847 15$:
      52    53   C0  0491   848          ADDL2   R3,R2                      ; Random number
      86    52   90  0494   849          MOVB    R2,(R6)+                   ; Fill in the transmit buffer
        F7  54   F5  0497   850          SOBGTR  R4,15$                     ; Branch if more bytes to be filled
                        049A   851
                        049A   852          $SETIMR_S -                        ; Set up one minute timer prevent hung
                        049A   853                    DAYTIM = ONEMIN,-
                        049A   854                    ASTADR = TIME_ERR_OUT,-
                        049A   855                    REQIDT = #RW_TIME_ID
                        04AD   856
      58    10   D0  04AD   857          MOVL    #LIMIT,R8                  ; Loop 100 times for each msg length
                        04B0   858 20$:
                        04B0   859          $QIO_S -                           ; Have a read data message outstanding
                        04BC   860                  EFN = #RECV_EFN,-
                        04B0   861                  CHAN = XM_CHAN,-
                        04B0   862                  FUNC = #IO$_READVBLK,-
                        04B0   863                  IOSB = RECV_IOSB,-
                        04B0   864                  ASTADR = RECV_AST,-
                        04B0   865                  ASTPRM = R7,-
                        04B0   866                  P1 = RECV_BUF,-
                        04B0   867                  P2 = R7
                        04D7   868
                        04D7   869          $QIOW_S -                          ; Transmit data message
                        04D7   870                  EFN = #XMIT_EFN,-
                        04D7   871                  CHAN = XM_CHAN,-
```

```
                        04D7   872                 FUNC = #IO$_WRITEVBLK,-
                        04D7   873                 IOSB = XM_IOSB,-
                        04D7   874                 P1 = XMIT_BUF,-
                        04D7   875                 P2 = R7
                        04FA   876
                 026B   30      04FA   877         BSBW    CHECK_IOSB              ; Check IO status block
                        04FD   878
                        04FD   879         $WAITFR_S EFN = #RECV_EFN               ; Wait until data received
                        0506   880
             0166'CF    D6      0506   881         INCL    ITERATION               ; Increment iteration count
                A3 58   F5      050A   882         SOBGTR  R8,20$                  ; Loop for 10 times
                        050D   883
                        050D   884         $CANTIM_S -                             ; Cancel hung timer
                        050D   885                 REQIDT = #RW_TIME_ID
                        0518   886
          0002'CF   02  B3      0518   887         BITW    #TEST_OVERM,FLAG        ; Is the test over?
                    09  12      051D   888         BNEQ    ATTN_MBX_TEST           ; BR if yes
                 03 57  F5      051F   889         SOBGTR  R7,30$                  ; For different message length
                  FF56  31      0522   890         BRW     XMIT_RECV               ; Try again
                  FF61  31      0525   891 30$:    BRW     10$
                        0528   892 ;
                        0528   893 ; Introduce an attention condition to see if attention AST delivered and mailbox
                        0528   894 ; receive appropriate message.
                        0528   895 ;
                        0528   896 ATTN_MBX_TEST:
                        0528   897
                        0528   898         $SETIMR_S -                             ; Set up one minute timer to prevent hung
                        0528   899                 DAYTIM = ONEMIN,-
                        0528   900                 ASTADR = TIME_ERR_OUT,-
                        0528   901                 REQIDT = #TIME_ID_2
                        053B   902
       03 0002'CF   04  E1      053B   903         BBC     #MODE_IS_ONEV,FLAG,10$  ; Br if mode is not "ONE"
                  0084  31      0541   904         BRW     CLEAN_EXIT
                        0544   905 10$:
                        0544   906         $QIO_S -                                ; Have an outstanding read mailbox message
                        0544   907                 CHAN = MBXCHAN,-
                        0544   908                 FUNC = #IO$_READVBLK,-
                        0544   909                 IOSB = XM_IOSB,-
                        0544   910                 ASTADR = CHK_MBX_AST,-
                        0544   911                 P1 = MBX_BUF,-
                        0544   912                 P2 = #MBXSIZE
                        056F   913
          0002'CF   20  A8      056F   914         BISW2   #TEST_ERRM,FLAG         ; Set flag say it's error test
                        0574   915
                        0574   916         $QIO_S -                                ; Send message without read request outstand
                        0574   917                 CHAN = XM_CHAN,-
                        0574   918                 FUNC = #IO$_WRITEVBLK,-
                        0574   919                 IOSB = XM_IOSB,-
                        0574   920                 P1 = XMIT_BUF,-
                        0574   921                 P2 = #128
                        059B   922
01A1'CF   000000C0 8F   D0      059B   923         MOVL    #MBXAST_DELM!ATTN_DELM,EF_MASK ; Set up mask for EFN wait
                        05A4   924
                        05A4   925         $WFLAND_S EFN = #MBXAST_DELV,-          ; Wait for MBX AST and ATTN AST delivered
                        05A4   926                 MASK = EF_MASK
                        05B1   927
                        05B1   928         $CLREF_S EFN = #MBXAST_DELV
```

```
                                05BA   929
                                05BA   930          $CLREF_S EFN = #ATTN_DELV
                                05C3   931
                0002'CF    20   AA   05C3   932          BICW2   #TEST_ERRM,FLAG                 ; Clear error test flag
                                05C8   933
                                05C8   934  CLEAN_EXIT:
                                05C8   935
                                05C8   936          $QIOW_S -                                   ; Disable attention AST
                                05C8   937                  CHAN = XM_CHAN,-
                                05C8   938                  FUNC = #IO$_SETMODE!IOSM_ATTNAST,-
                                05C8   939                  IOSB = XM_IOSB,-
                                05C8   940                  P1 = 0
                                05E9   941
                      017C      30   05E9   942          BSBW    CHECK_IOSB                      ; Check IO status block
                                05EC   943
                0002'CF 0040 8F   AA   05EC   944          BICW2   #FLAG_SHUTDNM,FLAG              ; Clear the shutdown flag
                                05F3   945
                                05F3   946          $QIOW_S -                                   ; Shut down the device
                                05F3   947                  CHAN = XM_CHAN,-
                                05F3   948                  FUNC = #IO$_SETMODE!IOSM_SHUTDOWN,-
                                05F3   949                  IOSB = XM_IOSB,-
                                05F3   950                  P1 = 0
                                0614   951
                      0151      30   0614   952          BSBW    CHECK_IOSB                      ; Check IO status block
                                0617   953
                                0617   954          $CANTIM_S REQIDT = #TIME_ID_2               ; Cancel timer
                                0622   955
                                0622   956  SUC_EXIT:
                                0622   957          $TRNLOG_S LOGNAM = MODE,-
                                0622   958                  RSLLEN = BUFFER_PTR,-
                                0622   959                  RSLBUF = FAO_BUF                   ; Get the run mode
                  0014'CF    20   8A   063B   960          BICB2   #LC_BITM,BUFFER                ; Convert to upper case
                0014'CF    4C 8F   91   0640   961          CMPB    #^A/L/,BUFFER                  ; Is this a loop for ever?
                           43      12   0646   962          BNEQ    10$                            ; BR if not
                0002'CF    02      AA   0648   963          BICW2   #TEST_OVERM,FLAG               ; Reset the termination flag
                      016A'CF      D6   064D   964          INCL    PASS                           ; Bump the pass count
                                0651   965          $FAO_S  CTRSTR = PASS_MSG,-
                                0651   966                  OUTLEN = BUFFER_PTR,-
                                0651   967                  OUTBUF = FAO_BUF,-
                                0651   968                  P1     = PASS,-
                                0651   969                  P2     = ITERATION,-
                                0651   970                  P3     = #0                           ; Make the end of pass message
                      000C'CF      DF   066E   971          PUSHAL  BUFFER_PTR                     ; Push the string desc.
                           01      DD   0672   972          PUSHL   #1                             ; Push arg count
                  00741133 8F      DD   0674   973          PUSHL   #UETP$_TEXT!STS$K_INFO         ; Push the signal name
                00000000'GF    03   FB   067A   974          CALLS   #3,G^LIB$SIGNAL                ; Print the end of pass message
                      0166'CF      D4   0681   975          CLRL    ITERATION                      ; Reset the iteration count
                         0025      30   0685   976          BSBW    START_DEV                      ; Restart the DMC/DMR
                         FD7F      31   0688   977          BRW     START_TEST                     ; Do the next pass
                                0688   978  10$:
56    0638'CF    00000638'8F      C1   0688   979          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block list header
                           02      88   0695   980          BISB2   #UETUNT$M_TESTABLE,-
                           08 A6        0697   981                  UETUNT$B_FLAGS(R6)     ; Set the testable bit
          0146'CF    10000001 8F      D0   0699   982          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                                06A2   983          $EXIT_S STATUS                         ; Exit with the status
                                06AD   984
```

```
                        06AD   986          .SBTTL  STARTDEV - Assign channel and start the device
                        06AD   987 ;++
                        06AD   988 ; FUNCTIONAL DESCRIPTION:
                        06AD   989 ;       This routine assigns channel, mailbox and start the device
                        06AD   990 ;
                        06AD   991 ; CALLING SEQUENCE:
                        06AD   992 ;       BSBW    START_DEV
                        06AD   993 ;
                        06AD   994 ; INPUT PARAMETERS:
                        06AD   995 ;       NONE
                        06AD   996 ;
                        06AD   997 ; IMPLICIT INPUTS:
                        06AD   998 ;       NONE
                        06AD   999 ;
                        06AD  1000 ; OUTPUT PARAMETERS:
                        06AD  1001 ;       NONE
                        06AD  1002 ;
                        06AD  1003 ; IMPLICIT OUTPUTS:
                        06AD  1004 ;       Exit with status if error
                        06AD  1005 ;
                        06AD  1006 ; COMPLETION CODES:
                        06AD  1007 ;       Error code of system service if error
                        06AD  1008 ;
                        06AD  1009 ; SIDE EFFECTS:
                        06AD  1010 ;       Program exit if error
                        06AD  1011 ;
                        06AD  1012 ;--
                        06AD  1013 START_DEV:
                        06AD  1014          $CREMBX_S -                     ; Create and assign channel mailbox
                        06AD  1015                    CHAN = MBXCHAN,-
                        06AD  1016                    MAXMSG = #MBXSIZE,-
                        06AD  1017                    BUFQUO = #MBXSIZE,-
                        06AD  1018                    LOGNAM = XMMBX_DESC
                        06CE  1019
                        06CE  1020          $ASSIGN_S -                     ; Assign channel to the device
                        06CE  1021                    DEVNAM = DEVDSC,-
                        06CE  1022                    CHAN = XM_CHAN,-
                        06CE  1023                    MBXNAM = XMMBX_DESC
                        06E3  1024
          22 50   E8    06E3  1025          BLBS    R0,10$                  ; BR if no failure
0146'CF    50   D0    06E6  1026          MOVL    R0,STATUS               ; Save the failure status
     0146'CF   DD    06EB  1027          PUSHL   STATUS                  ; Push the error code...
     0146'CF   DD    06EF  1028          PUSHL   STATUS
     0098'CF   DF    06F3  1029          PUSHAL  DEVDSC                  ; ...and the device designation...
     000F'CF   DF    06F7  1030          PUSHAL  TEST_NAME               ; ...and the test name...
          03   DD    06FB  1031          PUSHL   #3                      ; ...and the arg count...
 0074819A 8F   DD    06FD  1032          PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
          06   DD    0703  1033          PUSHL   #6                      ; ...and the total argument count...
        0468   31    0705  1034          BRW     ERROR_EXIT              ; ...and bail out completely
                      0708  1035 10$:
                      0708  1036
53  019B'CF    DE    0708  1037          MOVAL   DEVCHAR_BLK+2,R3        ; Address for max msg length
83  0200 8F    B0    070D  1038          MOVW    #MAX_MSG_LEN,(R3)+      ; Maximum message length
    63   12    90    0712  1039          MOVB    #XM$M_CHR_LOOPB!XM$M_CHR_MBX,(R3) ; Set loop back mode in char and
                      0715  1040                                          ; enable the associated mailbox
                      0715  1041          $SETIMR_S -                     ; Set up one minute timer to prevent hung
                      0715  1042                    DAYTIM = ONEMIN,-
```

```
                     0715  1043                ASTADR = TIME_ERR_OUT,-
                     0715  1044                REQIDT = #TIME_ID_1
                     0728  1045
                     0728  1046          $QIOW_S -                           ; Start the device
                     0728  1047                CHAN = XM_CHAN,-
                     0728  1048                FUNC = #IO$_SETMODE!IO$M_STARTUP,-
                     0728  1049                IOSB = XM_IOSB,-
                     0728  1050                ASTADR = CHK_QIO_AST,-
                     0728  1051                ASTPRM = #PRM,-
                     0728  1052                P1 = DEVCHAR_BLK,-
                     0728  1053                P3 = #1
                     0755  1054
                     0755  1055          $CANTIM_S REQIDT = #TIME_ID_1     ; Cancel timer
0002'CF   0040 BF    AB   0760  1056          BISW2   #FLAG_SHUTDNM,FLAG       ; Set flag to say shut down the
                     0767  1057                                            ; device if errors occur
                     05    0767  1058          RSB
```

```
                     0768  1060
                     0768  1061          .SBTTL  CHECKIOSB - Check IO status block
                     0768  1062 ;++
                     0768  1063 ; FUNCTIONAL DESCRIPTION:
                     0768  1064 ;       This routine checks the IO status block = #SS$_NORMAL
                     0768  1065 ;
                     0768  1066 ; CALLING SEQUENCE:
                     0768  1067 ;       BSBW    CHECK_IOSB
                     0768  1068 ;
                     0768  1069 ; INPUT PARAMETERS:
                     0768  1070 ;       NONE
                     0768  1071 ;
                     0768  1072 ; IMPLICIT INPUTS:
                     0768  1073 ;       NONE
                     0768  1074 ;
                     0768  1075 ; OUTPUT PARAMETERS:
                     0768  1076 ;       NONE
                     0768  1077 ;
                     0768  1078 ; IMPLICIT OUTPUTS:
                     0768  1079 ;       Exit with status if IOSB not right
                     0768  1080 ;
                     0768  1081 ; COMPLETION CODES:
                     0768  1082 ;       IO status in STATUS if error
                     0768  1083 ;
                     0768  1084 ; SIDE EFFECTS:
                     0768  1085 ;       Program exit if error found
                     0768  1086 ;--
                     0768  1087 ;--
                     0768  1088 CHECK_IOSB:
01   01A5'CF    B1   0768  1089          CMPW    XM_IOSB,#SS$_NORMAL     ; Is the QIO O.K.?
           01   12   076D  1090          BNEQ    10$                     ; Br if not
                05   076F  1091          RSB                             ; Return
                     0770  1092 10$:
7E   01A5'CF    3C   0770  1093          MOVZWL  XM_IOSB,-(SP)           ; Push the error status code
0146'CF    6E   D0   0775  1094          MOVL    (SP),STATUS             ; Set return status
           01   DD   077A  1095          PUSHL   #1                      ; Argument count
         03F1   31   077C  1096          BRW     ERROR_EXIT              ; Error exit
                     077F  1097
```

```
                                077F  1099          .SBTTL  Check Start Unit and Attention AST QIO AST Routine
                                077F  1100 ;++
                                077F  1101 ; FUNCTIONAL DESCRIPTION:
                                077F  1102 ;       This routine will be called as AST routine when QIO for start unit
                                077F  1103 ;       or attention AST is completed
                                077F  1104 ;       It checks IO status block and the AST parameter
                                077F  1105 ;
                                077F  1106 ; CALLING SEQUENCE:
                                077F  1107 ;       Called via AST at $QIO SETMODE!STARTUP or SETMODE!ATTNAST
                                077F  1108 ;
                                077F  1109 ; INPUT PARAMETERS:
                                077F  1110 ;       NONE
                                077F  1111 ;
                                077F  1112 ; IMPLICIT INPUTS:
                                077F  1113 ;       NONE
                                077F  1114 ;
                                077F  1115 ; OUTPUT PARAMETERS:
                                077F  1116 ;       NONE
                                077F  1117 ;
                                077F  1118 ; IMPLICIT OUTPUTS:
                                077F  1119 ;       Error message if error
                                077F  1120 ;
                                077F  1121 ; COMPLETION CODES:
                                077F  1122 ;       IO status in STATUS if error
                                077F  1123 ;
                                077F  1124 ; SIDE EFFECTS:
                                077F  1125 ;       Program exit if error
                                077F  1126 ;
                                077F  1127 ;--
                                077F  1128 CHK_QIO_AST:
                         0FFC   077F  1129          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                         FFE4    30  0781  1130          BSBW    CHECK_IOSB                      ; Go check IO status block
04 AC                00000064  8F   D1  0784  1131          CMPL    #PRM,4(AP)                      ; Check AST parameter
                           01    12  078C  1132          BNEQ    10$                             ; Branch if not #1 (STARTUP)
                                 04  078E  1133          RET
                                078F  1134 10$:
                      0293'CF    DF  078F  1135          PUSHAL  ASTPAR_ERRMSG                   ; Error message
                           01    DD  0793  1136          PUSHL   #1                              ; Arg count
                     00741132  8F    DD  0795  1137          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Signal name
                    0146'CF    6E    D0  079B  1138          MOVL    (SP),STATUS                     ; Set up status
                           03    DD  07A0  1139          PUSHL   #3                              ; Arg count
                         03CB    31  07A2  1140          BRW     ERROR_EXIT                      ; Error exit
```

```
                        07A5  1142          .SBTTL  Receive data AST routine
                        07A5  1143 ;++
                        07A5  1144 ; FUNCTIONAL DESCRIPTION:
                        07A5  1145 ;       This routine will be called as receive data AST routine
                        07A5  1146 ;       It checks IO status and compare the data in the receive buffer
                        07A5  1147 ;       against the transmit buffer
                        07A5  1148 ;
                        07A5  1149 ; CALLING SEQUENCE:
                        07A5  1150 ;       Called via AST at $QIO READ
                        07A5  1151 ;
                        07A5  1152 ; INPUT PARAMETERS:
                        07A5  1153 ;       AST parameter = message length
                        07A5  1154 ;
                        07A5  1155 ; IMPLICIT INPUTS:
                        07A5  1156 ;       NONE
                        07A5  1157 ;
                        07A5  1158 ; OUTPUT PARAMETERS:
                        07A5  1159 ;       NONE
                        07A5  1160 ;
                        07A5  1161 ; IMPLICIT OUTPUTS:
                        07A5  1162 ;       Error message if error found
                        07A5  1163 ;
                        07A5  1164 ; COMPLETION CODES:
                        07A5  1165 ;       in STATUS if error
                        07A5  1166 ;
                        07A5  1167 ; SIDE EFFECTS:
                        07A5  1168 ;       Program exit if error found
                        07A5  1169 ;
                        07A5  1170 ;--
                        07A5  1171 RECV_AST:
                   0FFC 07A5  1172          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
01   01AD'CF    B1   07A7  1173          CMPW    RECV_IOSB,#SS$_NORMAL   ; Is the read successful?
           15   12   07AC  1174          BNEQ    10$                     ; Br if not
  54     04 AC   D0   07AE  1175          MOVL    4(AP),R4                ; Message length
55   03B5'CF    DE   07B2  1176          MOVAL   RECV_BUF,R5             ; Address of receive buffer
56   01B5'CF    DE   07B7  1177          MOVAL   XMIT_BUF,R6             ; Address of transmit buffer
66   65   54    29   07BC  1178          CMPC3   R4,(R5),(R6)            ; Compare the data
           10   12   07C0  1179          BNEQ    20$                     ; Br if data not match
                04   07C2  1180          RET                             ; Return
                        07C3  1181 10$:
7E   01AD'CF    3C   07C3  1182          MOVZWL  RECV_IOSB,-(SP)         ; Push the error status code
0146'CF    6E   D0   07C8  1183          MOVL    (SP),STATUS             ; Set return status
           01   DD   07CD  1184          PUSHL   #1                      ; Argument count
         039E   31   07CF  1185          BRW     ERROR_EXIT              ; Error exit
                        07D2  1186 20$:
0635'CF    61   90   07D2  1187          MOVB    (R1),BAD_DATA           ; Bad data in the receive buffer
0636'CF    63   90   07D7  1188          MOVB    (R3),GOOD_DATA          ; The data in the transmit buffer
                        07DC  1189          $FAO_S -                        ; Format the output message
                        07DC  1190                  CTRSTR = RECV_ERR_MSG,-
                        07DC  1191                  OUTLEN = BUFFER_PTR,-
                        07DC  1192                  OUTBUF = FAO_BUF,-
                        07DC  1193                  P1 = GOOD_DATA,-
                        07DC  1194                  P2 = BAD_DATA
     000C'CF    DF   07F7  1195          PUSHAL  BUFFER_PTR              ; Push the string desc.
           01   DD   07FB  1196          PUSHL   #1                      ; Push arg count
 00741132  8F   DD   07FD  1197          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Push the signal name
0146'CF    6E   D0   0803  1198          MOVL    (SP),STATUS             ; Exit status
```

```
                                                M 10
                03   DD  0808  1199          PUSHL   #3                      ; Parameter count
          0363  31   31  080A  1200          BRW     ERROR_EXIT              ; Error exit
```

```
                        080D  1202          .SBTTL  Check mailbox message AST Routine
                        080D  1203 ;++
                        080D  1204 ; FUNCTIONAL DESCRIPTION:
                        080D  1205 ;       This routine will be called as AST routine when QIO for read mailbox
                        080D  1206 ;       is completed
                        080D  1207 ;       It checks IO status block and check message type in the mailbox when
                        080D  1208 ;       doing error test
                        080D  1209 ;
                        080D  1210 ; CALLING SEQUENCE:
                        080D  1211 ;       Called via AST at $QIO Read mailbox
                        080D  1212 ;
                        080D  1213 ; INPUT PARAMETERS:
                        080D  1214 ;       NONE
                        080D  1215 ;
                        080D  1216 ; IMPLICIT INPUTS:
                        080D  1217 ;       NONE
                        080D  1218 ;
                        080D  1219 ; OUTPUT PARAMETERS:
                        080D  1220 ;       NONE
                        080D  1221 ;
                        080D  1222 ; IMPLICIT OUTPUTS:
                        080D  1223 ;       NONE
                        080D  1224 ;
                        080D  1225 ; COMPLETION CODES:
                        080D  1226 ;       STATUS if error
                        080D  1227 ;
                        080D  1228 ; SIDE EFFECTS:
                        080D  1229 ;       Program exit if error
                        080D  1230 ;
                        080D  1231 ;--
                        080D  1232 CHK_MBX_AST:
                   0FFC 080D  1233          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                FF56 30 080F  1234          BSBW    CHECK_IOSB                  ; Check IO status block
 11 0002'CF      05 E1 0812  1235          BBC     #TEST_ERRV,FLAG,10$         ; Br if not intended error test
    05B5'CF      0B B1 0818  1236          CMPW    #MSG$_XM_DATAVL,MBX_BUF     ; Do we have right message type?
                35 12 081D  1237          BNEQ    20$                         ; Br if not
                      081F  1238          $SETEF_S EFN = #MBXAST_DELV         ; Set event flag say mailbox delivered
                04    0828  1239          RET                                 ; Return
                      0829  1240 10$:
                      0829  1241          $QIO_S -                            ; Have an outstanding read mailbox message
                      0829  1242                  CHAN = MBXCHAN,-
                      0829  1243                  FUNC = #IO$_READVBLK,-
                      0829  1244                  IOSB = XM_IOSB,-
                      0829  1245                  ASTADR = CHK_MBX_AST,-
                      0829  1246                  P1 = MBX_BUF,-
                      0829  1247                  P2 = #MBXSIZE
                      0853  1248
                04    0853  1249          RET
                      0854  1250 20$:
             02D0'CF  DF    0854  1251          PUSHAL  MBX_ERRMSG                  ; Set up the MBX error message
                   01 DD    0858  1252          PUSHL   #1                          ; Argument count
            00741132 8F     DD 085A  1253          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; Signal name
        0146'CF      6E     3C 0860  1254          MOVZWL  (SP),STATUS                 ; Set return status
                   03 DD    0865  1255          PUSHL   #3                          ; Argument count
                0306  31    0867  1256          BRW     ERROR_EXIT                  ; Error exit
                      086A  1257
```

```
                086A  1259          .SBTTL  Attention AST routine
                086A  1260 ;++
                086A  1261 ; FUNCTIONAL DESCRIPTION:
                086A  1262 ;       This routine will be called when the driver sets/clears
                086A  1263 ;       error summary bits or device status bits or data available but
                086A  1264 ;       no waiting read request
                086A  1265 ;       In error test, It sets a EF to indicate the AST delivered
                086A  1266 ;
                086A  1267 ; CALLING SEQUENCE:
                086A  1268 ;       Called via AST at $QIO SETMODE!ATTNAST
                086A  1269 ;
                086A  1270 ; INPUT PARAMETERS:
                086A  1271 ;       NONE
                086A  1272 ;
                086A  1273 ; IMPLICIT INPUTS:
                086A  1274 ;       NONE
                086A  1275 ;
                086A  1276 ; OUTPUT PARAMETERS:
                086A  1277 ;       NONE
                086A  1278 ;
                086A  1279 ; IMPLICIT OUTPUTS:
                086A  1280 ;       Error message if error
                086A  1281 ;
                086A  1282 ; COMPLETION CODES:
                086A  1283 ;       STATUS if error
                086A  1284 ;
                086A  1285 ; SIDE EFFECTS:
                086A  1286 ;       Program exit if error
                086A  1287 ;
                086A  1288 ;--
                086A  1289 XM_ATTN_AST:
          0FFC  086A  1290          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
59 0002'CF  05  E1  086C  1291          BBC     #TEST_ERRV,FLAG,10$     ; Br if not intended error test
                0872  1292
                0872  1293          $SETEF_S EFN = #ATTN_DELV        ; Set EF say attention AST delivered
                087B  1294
                087B  1295          $QIOW_S -                        ; Read the data message sent in error test
                087B  1296                  CHAN = XM_CHAN,-
                087B  1297                  FUNC = #IO$_READVBLK,-
                087B  1298                  IOSB = XM_IOSB,-
                087B  1299                  P1 = RECV_BUF,-
                087B  1300                  P2 = #128
                08A2  1301
          FEC3  30  08A2  1302          BSBW    CHECK_IOSB               ; Check IOSB
                08A5  1303
                08A5  1304          $QIOW_S -                        ; Enable attention AST, It's one shot
                08A5  1305                  CHAN = XM_CHAN,-
                08A5  1306                  FUNC = #IO$_SETMODE!IO$M_ATTNAST,-
                08A5  1307                  IOSB = XM_IOSB,-
                08A5  1308                  P1 = XM_ATTN_AST
          FE9E  30  08C7  1309          BSBW    CHECK_IOSB               ; Check IOSB
                04  08CA  1310          RET                              ; Return
```

```
                     08CB  1312 10$:                                  ; Check to see what's wrong
54   04 AC    D0     08CB  1313          MOVL    4(AP),R4               ; Dev characs are passed as args
27 54   10    E0     08CF  1314          BBS     #XM$V_ERR_FATAL,R4,15$ ; BR if fatal error
2A 54   14    E0     08D3  1315          BBS     #XM$V_ERR_LOST,R4,20$  ; BR if data lost error
2D 54   17    E0     08D7  1316          BBS     #XM$V_ERR_START,R4,25$ ; BR if DDCMP START message
30 54   13    E0     08DB  1317          BBS     #XM$V_ERR_MAINT,R4,30$ ; BR if DDCMP maintenance msg received
32 54   0A    E0     08DF  1318          BBS     #XM$V_STS_ORUN,R4,35$  ; BR if data overrun
35 54   08    E0     08E3  1319          BBS     #XM$V_STS_DCHK,R4,40$  ; BR if retransmission threshold excded
38 54   09    E0     08E7  1320          BBS     #XM$V_STS_TIMO,R4,45$  ; BR if DDCMP timeout
3B 54   0E    E0     08EB  1321          BBS     #XM$V_STS_DISC,R4,50$  ; BR if DISC error
3E 54   0B    E0     08EF  1322          BBS     #XM$V_STS_ACTIVE,R4,55$ ; BR if protocol still active
   04DB'CF    DF     08F3  1323          PUSHAL  ERR_ATTN_MSG           ; Something else
      009A    31     08F7  1324          BRW     70$
                     08FA  1325 15$:
   0304'CF    DF     08FA  1326          PUSHAL  ERR_FATAL_MSG          ; Error message
      0093    31     08FE  1327          BRW     70$
                     0901  1328 20$:
   033A'CF    DF     0901  1329          PUSHAL  ERR_LOST_MSG           ; Error message
      008C    31     0905  1330          BRW     70$
                     0908  1331 25$:
   037C'CF    DF     0908  1332          PUSHAL  ERR_START_MSG          ; ...
      0085    31     090C  1333          BRW     70$
                     090F  1334 30$:
   03AE'CF    DF     090F  1335          PUSHAL  ERR_MAINT_MSG          ; ...
         7F    11     0913  1336          BRB     70$
                     0915  1337 35$:
55 03E6'CF    DE     0915  1338          MOVAL   STS_ORUN_MSG,R5
         1A    11     091A  1339          BRB     65$
                     091C  1340 40$:
55 0424'CF    DE     091C  1341          MOVAL   STS_DCHK_MSG,R5
         13    11     0921  1342          BRB     65$
                     0923  1343 45$:
55 0459'CF    DE     0923  1344          MOVAL   STS_TIMO_MSG,R5
         0C    11     0928  1345          BRB     65$
                     092A  1346 50$:
55 046E'CF    DE     092A  1347          MOVAL   STS_DISC_MSG,R5
         05    11     092F  1348          BRB     65$
                     0931  1349 55$:
55 04A3'CF    DE     0931  1350          MOVAL   NO_WAIT_READ,R5
                     0936  1351 65$:
                     0936  1352          $QIO_S  CHAN = MBXCHAN,-       ; Read mailbox associated with attn msg
                     0936  1353                  FUNC = #IO$_READVBLK,-
                     0936  1354                  P1   = MBX_BUF,-
                     0936  1355                  P2   = #MBXSIZE
05B5'CF    02   39   095B  1356          MATCHC  #2,MBX_BUF,-                   ; Figure out...
0554'CF    08        0960  1357          #ATTN_MBX_TYPES_LENGTH,ATTN_MBX_TYPES
      52   02   C6   0964  1358          DIVL2   #2,R2                          ; ...just what kind...
           52   D6   0967  1359          INCL    R2
56 055C'CF42   D0   0969  1360          MOVL    ATTN_MBX_TYPES_NAMES[R2],R6 ; ...of mailbox this is
                     096F  1361          $FAO_S  CTRSTR = ATTN_MBX_MSG,-
                     096F  1362                  OUTLEN = BUFFER_PTR,-
                     096F  1363                  OUTBUF = FAO_BUF,-
                     096F  1364                  P1     = R5,-
                     096F  1365                  P2     = R6,-
                     096F  1366                  P3     = #MBX_BUF+4,-
                     096F  1367                  P4     = MBX_BUF+2
   000C'CF    DF     0990  1368          PUSHAL  BUFFER_PTR
```

```
                     0994  1369 70$:
          01   DD    0994  1370          PUSHL   #1                              ; Argument count
00741132 8F   DD    0996  1371          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Error code
0146'CF   6E   D0    099C  1372          MOVL    (SP),STATUS                     ; Save in STATUS
          03   DD    09A1  1373          PUSHL   #3                              ; Argument count
        01CA   31    09A3  1374          BRW     ERROR_EXIT                      ; Error exit
                     09A6  1375
```

```
                     09A6  1377          .SBTTL  One Minute Timer Expiration Routine
                     09A6  1378 ;++
                     09A6  1379 ; FUNCTIONAL DESCRIPTION:
                     09A6  1380 ;       This routine will be called only if the timer which was set to prevent
                     09A6  1381 ;       program hangs goes off.
                     09A6  1382 ;
                     09A6  1383 ; CALLING SEQUENCE:
                     09A6  1384 ;       Called via AST at $SETIMR expiration.
                     09A6  1385 ;
                     09A6  1386 ; INPUT PARAMETERS:
                     09A6  1387 ;       NONE
                     09A6  1388 ;
                     09A6  1389 ; IMPLICIT INPUTS:
                     09A6  1390 ;       NONE
                     09A6  1391 ;
                     09A6  1392 ; OUTPUT PARAMETERS:
                     09A6  1393 ;       NONE
                     09A6  1394 ;
                     09A6  1395 ; IMPLICIT OUTPUTS:
                     09A6  1396 ;       NONE
                     09A6  1397 ;
                     09A6  1398 ; COMPLETION CODES:
                     09A6  1399 ;       NONE
                     09A6  1400 ;
                     09A6  1401 ; SIDE EFFECTS:
                     09A6  1402 ;       NONE
                     09A6  1403 ;
                     09A6  1404 ;--
                     09A6  1405
                     09A6  1406 TIME_ERR_OUT:
                0FFC 09A6  1407          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
 0000022C 8F     DD  09A8  1408          PUSHL   #SS$_TIMEOUT                        ; Push the signal name
0146'CF   6E     D0  09AE  1409          MOVL    (SP),STATUS                         ; Set exit status
          01     DD  09B3  1410          PUSHL   #1                                  ; Push the argument count total
        01B8     31  09B5  1411          BRW     ERROR_EXIT                          ; Bail out completely
```

```
                     09B8  1413          .SBTTL  Three Minutes Timer Expiration Routine
                     09B8  1414 ;++
                     09B8  1415 ; FUNCTIONAL DESCRIPTION:
                     09B8  1416 ;       This routine will be called when the device test has been run for
                     09B8  1417 ;       about three minutes.
                     09B8  1418 ;
                     09B8  1419 ; CALLING SEQUENCE:
                     09B8  1420 ;       Called via AST at $SETIMR expiration.
                     09B8  1421 ;
                     09B8  1422 ; INPUT PARAMETERS:
                     09B8  1423 ;       NONE
                     09B8  1424 ;
                     09B8  1425 ; IMPLICIT INPUTS:
                     09B8  1426 ;       NONE
                     09B8  1427 ;
                     09B8  1428 ; OUTPUT PARAMETERS:
                     09B8  1429 ;       NONE
                     09B8  1430 ;
                     09B8  1431 ; IMPLICIT OUTPUTS:
                     09B8  1432 ;       NONE
                     09B8  1433 ;
                     09B8  1434 ; COMPLETION CODES:
                     09B8  1435 ;       NONE
                     09B8  1436 ;
                     09B8  1437 ; SIDE EFFECTS:
                     09B8  1438 ;       Sets a flag to indicate timer expiration.
                     09B8  1439 ;
                     09B8  1440 ;--
                     09B8  1441
                     09B8  1442 TIME_SUC_OUT:
                0FFC 09B8  1443          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     09BA  1444
0002'CF   02    A8   09BA  1445          BISW2   #TEST_OVERM,FLAG                    ; set test over bit
                04   09BF  1446          RET
```

```
09C0  1448
09C0  1449          .SBTTL  System Service Exception Handler
09C0  1450 ;++
09C0  1451 ; FUNCTIONAL DESCRIPTION:
09C0  1452 ;       This routine is executed if a software or hardware exception occurs or
09C0  1453 ;       if a LIB$SIGNAL system service is used to output a message.
09C0  1454 ;
09C0  1455 ; CALLING SEQUENCE:
09C0  1456 ;       Entered via an exception from the system
09C0  1457 ;
09C0  1458 ; INPUT PARAMETERS:
09C0  1459 ;       ERROR_COUNT    = previous cumulative error count
09C0  1460 ;
09C0  1461 ;          AP ----->    +---------------+
09C0  1462 ;                       |       2       |
09C0  1463 ;                       +---------------+
09C0  1464 ;                       |  SIGNL ARY PNT|
09C0  1465 ;                       +---------------+          +---------+
09C0  1466 ;                       |  MECH  ARY PNT|          |         |
09C0  1467 ;                       +---------------+          |         ^
09C0  1468 ;                       |       4       |          |         |
09C0  1469 ;                       +---------------+          |         |
09C0  1470 ;                       |  ESTABLISH FP |          |         |
09C0  1471 ;                       +---------------+          | Mechanism Array
09C0  1472 ;                       |     DEPTH     |          |         |
09C0  1473 ;                       +---------------+          |         |
09C0  1474 ;                       |       R0      |          |         |
09C0  1475 ;                       +---------------+          |         v
09C0  1476 ;                       |       R1      |          +---------+
09C0  1477 ;                       +---------------+          |         ^
09C0  1478 ;                       |       N       |          |         |
09C0  1479 ;                       +---------------+          |         |
09C0  1480 ;                       |  CONDITION NAME|         |         |
09C0  1481 ;                       +---------------+          |         |
09C0  1482 ;                       | N-3 ADDITIONAL|          | Signal Array
09C0  1483 ;                       | LONG WORD ARGS|          |         |
09C0  1484 ;                       +---------------+          |         |
09C0  1485 ;                       |       PC      |          |         |
09C0  1486 ;                       +---------------+          |         |
09C0  1487 ;                       |       PSL     |          |         v
09C0  1488 ;                       +---------------+          +---------+
09C0  1489 ; IMPLICIT INPUTS:
09C0  1490 ;       NONE
09C0  1491 ;
09C0  1492 ; OUTPUT PARAMETERS:
09C0  1493 ;       NONE
09C0  1494 ;
09C0  1495 ; IMPLICIT OUTPUTS:
09C0  1496 ;       NONE
09C0  1497 ;
09C0  1498 ; COMPLETION CODES:
09C0  1499 ;       SS$_NORMAL if it's a UETP condition or RMS error.
09C0  1500 ;       Error status from exception, otherwise.
09C0  1501 ;
09C0  1502 ; SIDE EFFECTS:
09C0  1503 ;       May branch to ERROR_EXIT.
09C0  1504 ;       May print a message.
09C0  1505 ;--
```

```
                                09C0  1505
                                09C0  1506 SSERROR:
                           0FFC 09C0  1507         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                09C2  1508
                                09C2  1509         $SETAST_S ENBFLG = #0                ; Disable AST delivery
                         01     DD  09CB  1510         PUSHL   #1                   ; Assume ASTs were enabled
                    50   09     D1  09CD  1511         CMPL    S^#SS$_WASSET,R0     ; Were ASTs enabled?
                         02     13  09D0  1512         BEQL    10$                  ; BR if they were
                         6E     D4  09D2  1513         CLRL    (SP)                 ; Set ASTs to remain disabled
                                09D4  1514 10$:
                                09D4  1515         $SETSFM_S ENBFLG = #0                ; Disable SS failure mode
                         01     DD  09DD  1516         PUSHL   #1                   ; Assume SS failure mode was enabled
                    50   09     D1  09DF  1517         CMPL    S^#SS$_WASSET,R0     ; Was SS failure mode enabled?
                         02     13  09E2  1518         BEQL    20$                  ; BR if it was
                         6E     D4  09E4  1519         CLRL    (SP)                 ; Set SS failure mode to remain off
                                09E6  1520 20$:
                56   04 AC     D0  09E6  1521         MOVL    CHF$L_SIGARGLST(AP),R6 ; Get the signal array pointer
                59   04 A6     7D  09EA  1522         MOVQ    CHF$L_SIG_NAME(R6),R9 ; Get NAME in R9 and ARG1 in R10
                           10    ED  09EE  1523         CMPZV   #STS$V_FAC_NO,- ; Is this a message from LIB$SIGNAL?
                           0C        09F0  1524                 #STS$S_FAC_NO,-
                00000074   8F     59  09F1  1525                 R9,#UETP$_FACILITY
                           14     12  09F7  1526         BNEQ    30$                  ; BR if this is not a UETP exception
                      66   02     C2  09F9  1527         SUBL2   #2,CHF$L_SIG_ARGS(R6) ; Drop the PC and PSL
                                09FC  1528         $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                           21     11  0A0B  1529         BRB     40$                  ; Restore ASTs and SS fail mode
                                0A0D  1530 30$:
           59   0000045C      8F     D1  0A0D  1531         CMPL    #SS$_SSFAIL,R9       ; RMS failures are SysSvc failures
                           32     12  0A14  1532         BNEQ    50$                  ; BR if this can't be an RMS failure
                           10     ED  0A16  1533         CMPZV   #STS$V_FAC_NO,- ; Is it an RMS failure?
                           0C        0A18  1534                 #STS$S_FAC_NO,-
                      01   5A        0A19  1535                 R10,#RMS$_FACILITY
                           2B     12  0A1B  1536         BNEQ    50$                  ; BR if not
           5A   F0000000   8F     CA  0A1D  1537         BICL2   #^XF0000000,R10       ; Strip control bits from status code
                 08 A6     04     39  0A24  1538         MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                           14        0A28  1539                 #NRAT_LENGTH,-
                004D'CF              0A29  1540                 NO_RMS_AST_TABLE     ; ...no AST can be delivered?
                           1A     13  0A2C  1541         BEQL    50$                  ; BR if so - must give error here
                                0A2E  1542 40$:
                           01     BA  0A2E  1543         POPR    #^M<R0>              ; Restore SS failure mode...
                                0A30  1544         $SETSFM_S ENBFLG = R0                ; ...
                           01     BA  0A39  1545         POPR    #^M<R0>              ; Restore AST enable...
                                0A3B  1546         $SETAST_S ENBFLG = R0                ; ...
                      50   01     D0  0A44  1547         MOVL    S^#SS$_NORMAL,R0     ; Supply a standard status for exit
                           04     04  0A47  1548         RET                          ; Resume processing (or goto RMS_ERROR)
                                0A48  1549 50$:
                0146'CF    59     D0  0A48  1550         MOVL    R9,STATUS            ; Save the status
                           58     D4  0A4D  1551         CLRL    R8                   ; Assume for now it's not SS failure
           59   0000045C      8F     D1  0A4F  1552         CMPL    #SS$_SSFAIL,R9       ; But is it a System Service failure?
                           38     12  0A56  1553         BNEQ    70$                  ; BR if not - no special case message
                                0A58  1554         $GETMSG_S MSGID = R10,-      ; Get SS failure code associated text
                                0A58  1555                   MSGLEN = BUFFER_PTR,-
                                0A58  1556                   BUFADR = FAO_BUF,-
                                0A58  1557                   FLAGS  = #14,-
                                0A58  1558                   OUTADR = MSG_BLOCK
                016F'CF    95     0A6F  1559         TSTB    MSG_BLOCK+1          ; Get FAO arg count for SS failure code
                           16     13  0A73  1560         BEQL    60$                  ; Don't use $GETMSG if no $FAO args...
                000C'CF    DF     0A75  1561         PUSHAL  BUFFER_PTR           ; ...else build up...
```

```
                01   DD  0A79  1562                PUSHL   #1                              ; ...a message describing...
          00741130  8F   DD  0A7B  1563                PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                00   5A   F0  0A81  1564                INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                6E   03        0A84  1565                        #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                58   03   D0  0A86  1566                MOVL    #3,R8                           ; Count the number of args we pushed
                     05   11  0A89  1567                BRB     70$
                          0A8B  1568 60$:
                     5A   DD  0A8B  1569                PUSHL   R10                             ; Save SS failure code
                     58   01  D0  0A8D  1570                MOVL    #1,R8                           ; Count the number of args we pushed
                          0A90  1571 70$:
           57   66   04   C5  0A90  1572                MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                5E   57   C2  0A94  1573                SUBL2   R7,SP                           ; Save the current signal array...
        6E   04 A6   57   28  0A97  1574                MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
           7E   66   58   C1  0A9C  1575                ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                     00CD   31  0AA0  1576                BRW     ERROR_EXIT
```

```
                                                J 11
                        0AA3  1578          .SBTTL  RMS Error Handler
                        0AA3  1579 ;++
                        0AA3  1580 ; FUNCTIONAL DESCRIPTION:
                        0AA3  1581 ;       This routine handles error returns from RMS calls.
                        0AA3  1582 ;
                        0AA3  1583 ; CALLING SEQUENCE:
                        0AA3  1584 ;       Called by RMS when a file processing error is found.
                        0AA3  1585 ;
                        0AA3  1586 ; INPUT PARAMETERS:
                        0AA3  1587 ;       The FAB or RAB associated with the RMS call.
                        0AA3  1588 ;
                        0AA3  1589 ; IMPLICIT INPUTS:
                        0AA3  1590 ;       NONE
                        0AA3  1591 ;
                        0AA3  1592 ; OUTPUT PARAMETERS:
                        0AA3  1593 ;       NONE
                        0AA3  1594 ;
                        0AA3  1595 ; IMPLICIT OUTPUTS:
                        0AA3  1596 ;       Error message
                        0AA3  1597 ;
                        0AA3  1598 ; COMPLETION CODES:
                        0AA3  1599 ;       NONE
                        0AA3  1600 ;
                        0AA3  1601 ; SIDE EFFECTS:
                        0AA3  1602 ;       Program may exit, depending on severity of the error.
                        0AA3  1603 ;
                        0AA3  1604 ;--
                        0AA3  1605
                        0AA3  1606 RMS_ERROR:
                  0FFC  0AA3  1607          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0AA5  1608
          56  04 AC    D0  0AA5  1609          MOVL    4(AP),R6                ; See whether we're dealing with...
              66  03    91  0AA9  1610          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                  16    12  0AAC  1611          BNEQ    10$                     ; BR if it's a RAB
          57  01FD'CF    DE  0AAE  1612          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
              58  56    D0  0AB3  1613          MOVL    R6,R8                   ; ...address of FAB...
                0C  A6    DD  0AB6  1614          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
                08  A6    DD  0AB9  1615          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
  0146'CF       08  A6    D0  0ABC  1616          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
                    15    11  0AC2  1617          BRB     COMMON                  ; FAB and RAB share other code
                        0AC4  1618 10$:
          57  0209'CF    DE  0AC4  1619          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
              58  3C    A6    D0  0AC9  1620          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
                0C  A6    DD  0ACD  1621          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
                08  A6    DD  0AD0  1622          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
  0146'CF       08  A6    D0  0AD3  1623          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                        0AD9  1624 COMMON:
          5A  34  AB    9A  0AD9  1625          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                        0ADD  1626          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                        0ADD  1627                  OUTLEN = BUFFER_PTR,-
                        0ADD  1628                  OUTBUF = FAO_BUF,-
                        0ADD  1629                  P1     = R7,-
                        0ADD  1630                  P2     = R10,-
                        0ADD  1631                  P3     = FAB$L_FNA(R8)
              000C'CF   DF  0AF7  1632          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
                    01  DD  0AFB  1633          PUSHL   #1                      ; ...
          00741130  8F    DD  0AFD  1634          PUSHL   #UETP$_TEXT             ; ...
```

```
                00   EF  0B03  1635          EXTZV   #STS$V_SEVERITY,-
                03       0B05  1636                  #STS$S_SEVERITY,-
          59    0146'CF  0B06  1637                  STATUS,R9                   ; ...get the severity code...
                6E   59  88    0B0A  1638          BISB2   R9,(SP)              ; ...and add it into the signal name
                     05  DD    0B0D  1639          PUSHL   #5                   ; Current arg count
                   005E  31    0B0F  1640          BRW     ERROR_EXIT
```

```
                                0B12  1642          .SBTTL  CTRL/C Handler
                                0B12  1643 ;++
                                0B12  1644 ; FUNCTIONAL DESCRIPTION:
                                0B12  1645 ;       This routine handles CTRL/C AST's
                                0B12  1646 ;
                                0B12  1647 ; CALLING SEQUENCE:
                                0B12  1648 ;       Called via AST
                                0B12  1649 ;
                                0B12  1650 ; INPUT PARAMETERS:
                                0B12  1651 ;       NONE
                                0B12  1652 ;
                                0B12  1653 ; IMPLICIT INPUTS:
                                0B12  1654 ;       NONE
                                0B12  1655 ;
                                0B12  1656 ; OUTPUT PARAMETERS:
                                0B12  1657 ;       NONE
                                0B12  1658 ;
                                0B12  1659 ; IMPLICIT OUTPUTS:
                                0B12  1660 ;       NONE
                                0B12  1661 ;
                                0B12  1662 ; COMPLETION CODES:
                                0B12  1663 ;       NONE
                                0B12  1664 ;
                                0B12  1665 ; SIDE EFFECTS:
                                0B12  1666 ;       NONE
                                0B12  1667 ;
                                0B12  1668 ;--
                                0B12  1669
                                0B12  1670 CCASTHAND:
                        0FFC    0B12  1671          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                0B14  1672
     21 0002'CF    06    E1    0B14  1673          BBC     #FLAG_SHUTDNV,FLAG,10$  ; Have to shut down device?
                                0B1A  1674          $QIO_S -                        ; Shut down the device
                                0B1A  1675                  CHAN = XM_CHAN,-
                                0B1A  1676                  FUNC = #IO$_SETMODE!IO$M_SHUTDOWN,-
                                0B1A  1677                  IOSB = XM_IOSB,-
                                0B1A  1678                  P1 = 0
                                0B3B  1679 10$:
                00A3'CF    DF   0B3B  1680          PUSHAL  CNTRLCMSG               ; Set message pointer
                        01  DD   0B3F  1681          PUSHL   #1                      ; Set arg count
                00741130   8F   DD   0B41  1682          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                        00  DD   0B47  1683          PUSHL   #0                      ; Indicate an abnormal termination
                00A0'CF    DF   0B49  1684          PUSHAL  PROCESS_NAME            ; ...
                        02  DD   0B4D  1685          PUSHL   #2                      ; ...
                007410E0   8F   DD   0B4F  1686          PUSHL   #UETP$_ABENDD'STS$K_WARNING ; ...
           00000000'GF     07   FB   0B55  1687          CALLS   #7,G^LIB$SIGNAL         ; Output the message
                           D0   0B5C  1688          MOVL    #<STS$M_INHIB_MSG!-     ; Set the exit status
                                0B5D  1689                  SS$_CONTROLC-=
                                0B5D  1690                  STS$K_SUCCESS+STS$K_WARNING>,-
    0146'CF    10000650   8F        0B5D  1691                  STATUS
                                0B65  1692          $EXIT_S STATUS                  ; Terminate program cleanly
```

```
                                0B70  1694          .SBTTL  Error Exit
                                0B70  1695 ;++
                                0B70  1696 ; FUNCTIONAL DESCRIPTION:
                                0B70  1697 ;       This routine prints an error message and exits.
                                0B70  1698 ;
                                0B70  1699 ; CALLING SEQUENCE:
                                0B70  1700 ;       MOVx  error status value,STATUS
                                0B70  1701 ;       PUSHx error specific information on the stack
                                0B70  1702 ;       PUSHL current argument count
                                0B70  1703 ;       BRW   ERROR_EXIT
                                0B70  1704 ;
                                0B70  1705 ; INPUT PARAMETERS:
                                0B70  1706 ;       Arguments to LIB$SIGNAL, as above
                                0B70  1707 ;
                                0B70  1708 ; IMPLICIT INPUTS:
                                0B70  1709 ;       NONE
                                0B70  1710 ;
                                0B70  1711 ; OUTPUT PARAMETERS:
                                0B70  1712 ;       Message to SYS$OUTPUT and SYS$ERROR
                                0B70  1713 ;
                                0B70  1714 ; IMPLICIT OUTPUTS:
                                0B70  1715 ;       Program exit
                                0B70  1716 ;
                                0B70  1717 ; COMPLETION CODES:
                                0B70  1718 ;       Error in STATUS
                                0B70  1719 ;
                                0B70  1720 ; SIDE EFFECTS:
                                0B70  1721 ;       NONE
                                0B70  1722 ;
                                0B70  1723 ;--
                                0B70  1724
                                0B70  1725 ERROR_EXIT:
                                0B70  1726
                                0B70  1727          $SETAST_S ENBFLG = #0                ; ASTs can play havoc with messages
      15 0002'CF    03    E0    0B79  1728          BBS     #BEGIN_MSGV,FLAG,10$         ; BR if "begin" msg already printed
                       7E    D4  0B7F  1729          CLRL    -(SP)                        ; Set the time stamp flag
                 000F'CF    DF  0B81  1730          PUSHAL  TEST_NAME                    ; Set the test name
                       02    DD  0B85  1731          PUSHL   #2                           ; Push the argument count
                 00741039    8F  0B87  1732          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
           00000000'GF    04    FB  0B8D  1733          CALLS   #4,G^LIB$SIGNAL              ; Print the startup message
                                0B94  1734 10$:
          0182'CF    08    8E    C1  0B94  1735          ADDL3   (SP)+,#8,ARG_COUNT           ; Get total # args, pop partial count
                 0142'CF    D6  0B9A  1736          INCL    ERROR_COUNT                  ; Keep running error count
                       00    DD  0B9E  1737          PUSHL   #0                           ; Push the time parameter
                 00A0'CF    DF  0BA0  1738          PUSHAL  PROCESS_NAME                 ; Push test name...
               000F0002    8F  0BA4  1739          PUSHL   #^XF0002                     ; ...arg count...
               007410E2    8F  0BAA  1740          PUSHL   #UETP$_ABENDD!STS$K_ERROR    ; ...and signal name
                 0142'CF    DD  0BB0  1741          PUSHL   ERROR_COUNT                  ; Finish off arg list...
                 00A0'CF    DF  0BB4  1742          PUSHAL  PROCESS_NAME                 ; ...
               00010002    8F  0BB8  1743          PUSHL   #^X10002                     ; ...
               00748022    8F  0BBE  1744          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
     00000000'GF    0182'CF    FB  0BC4  1745          CALLS   ARG_COUNT,G^LIB$SIGNAL       ; Truly bitch
                                0BCD  1746
                 0146'CF    D5  0BCD  1747          TSTL    STATUS                       ; Did we exit with an error code?
                       09    12  0BD1  1748          BNEQ    20$                          ; BR if we did
               007410E2    8F  0BD3  1749          MOVL    #UETP$_ABENDD!STS$K_ERROR,-  ; Supply a generic one otherwise
                 0146'CF        0BD9  1750                  STATUS
```

```
                     0BDC  1751 20$:
          21 0002'CF   06   E1 0BDC  1752          BBC     #FLAG_SHUTDNV,FLAG,30$  ; Have to shut down device?
                     0BE2  1753          $QIO_S -                        ; Shut down the device
                     0BE2  1754                  CHAN = XM_CHAN,-
                     0BE2  1755                  FUNC = #IO$_SETMODE!IO$M_SHUTDOWN,-
                     0BE2  1756                  IOSB = XM_IOSB,-
                     0BE2  1757                  P1 = 0
                     0C03  1758 30$:
      0146'CF    10000000 8F   C8 0C03  1759          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                     0C0C  1760          $EXIT_S STATUS                  ; Exit in error
```

```
                        0C17  1762          .SBTTL  Exit Handler
                        0C17  1763 ;++
                        0C17  1764 ; FUNCTIONAL DESCRIPTION:
                        0C17  1765 ;       This routine handles cleanup at exit.  If the MODE logical name is
                        0C17  1766 ;       equated to "ONE", the routine will update the test flag in the
                        0C17  1767 ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                        0C17  1768 ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                        0C17  1769 ;       under test.
                        0C17  1770 ;
                        0C17  1771 ; CALLING SEQUENCE:
                        0C17  1772 ;       Invoked automatically by $EXIT System Service.
                        0C17  1773 ;
                        0C17  1774 ; INPUT PARAMETERS:
                        0C17  1775 ;       STATUS  contains the exit status.
                        0C17  1776 ;       FLAG    has synchronizing bits.
                        0C17  1777 ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                        0C17  1778 ;
                        0C17  1779 ; IMPLICIT INPUTS:
                        0C17  1780 ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                        0C17  1781 ;                 blocks for the device under test.
                        0C17  1782 ;
                        0C17  1783 ; OUTPUT PARAMETERS:
                        0C17  1784 ;       NONE
                        0C17  1785 ;
                        0C17  1786 ; IMPLICIT OUTPUTS:
                        0C17  1787 ;       Various files are de-accessed, the process name is reset, and any
                        0C17  1788 ;       necessary synchronization with UETPDEV01 is carried out.
                        0C17  1789 ;       If the MODE logical name is equated to "ONE", the routine will update
                        0C17  1790 ;       the test flag in the UETINIDEV.DAT file depending on the
                        0C17  1791 ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                        0C17  1792 ;       block for each unit for the device under test.
                        0C17  1793 ;
                        0C17  1794 ; COMPLETION CODES:
                        0C17  1795 ;       NONE
                        0C17  1796 ;
                        0C17  1797 ; SIDE EFFECTS:
                        0C17  1798 ;       NONE
                        0C17  1799 ;
                        0C17  1800 ;--
                        0C17  1801
                        0C17  1802 EXIT_HANDLER:
                  0FFC  0C17  1803          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0C19  1804
                        0C19  1805          $SETSFM_S ENBFLG = #0                   ; Turn off System Service failure mode
                        0C22  1806          $SETAST_S ENBFLG = #0                   ; No more ASTs
                        0C2B  1807          $TRNLOG_S LOGNAM = MODE,-               ; Get the run mode
                        0C2B  1808                    RSLLEN = BUFFER_PTR,-
                        0C2B  1809                    RSLBUF = FAO_BUF
   0014'CF    20   8A   0C44  1810          BICB2   #LC_BITM,BUFFER                 ; Convert to upper case
0014'CF    4F 8F   91   0C49  1811          CMPB    #^A/O/,BUFFER                   ; Is this a one shot?
            03   13   0C4F  1812          BEQL    10$                             ; BR if yes...
          00B8   31   0C51  1813          BRW     END_UPDATE                      ; ...else don't update UETINIDEV.DAT
                        0C54  1814 10$:
03 0002'CF    02   E0   0C54  1815          BBS     #SAFE_TO_UPDV,FLAG,20$          ; Only update if it's safe
          00AF   31   0C5A  1816          BRW     END_UPDATE                      ; Else forget it
                        0C5D  1817 20$:
   5A   072C'CF    DE   0C5D  1818          MOVAL   INI_RAB,R10                     ; Set the RAB address
```

```
                1E AA   02    90  0C62  1819          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
          10 AA 0770'CF 06    28  0C66  1820          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                  0C6D  1821          $GET    RAB = (R10)               ; Go back to the DDB record
                   75 50    E9  0C76  1822          BLBC    R0,UPDATE_FAILED          ; If failure then forget it
                1E AA   00    90  0C79  1823          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
     5B 0638'CF 00000638'8F   C1  0C7D  1824          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                      59    D4  0C87  1825          CLRL    R9                        ; Init a counter
                                  0C89  1826 UNIT_LOOP:
                      01    E1  0C89  1827          BBC     #UETUNT$V_TESTABLE,-      ; BR if this unit is not testable
                   02 0B AB       0C8B  1828                  UETUNT$B_FLAGS(R11),10$
                      59    D6  0C8E  1829          INCL    R9                        ; Count testable units
                                  0C90  1830 10$:
                   5B    6B    C0  0C90  1831          ADDL2   (R11),R11                 ; Next unit block
                00000638'8F 5B    D1  0C93  1832          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                      ED    12  0C9A  1833          BNEQ    UNIT_LOOP                 ; BR if not
                      59    D5  0C9C  1834          TSTL    R9                        ; Any testable units?
                      12    12  0C9E  1835          BNEQ    20$                       ; BR if yes...
                0018'CF 4E 8F   90  0CA0  1836          MOVB    #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                  0CA6  1837          $UPDATE RAB = (R10)               ; ...here
                      3C 50    E9  0CAF  1838          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                                  0CB2  1839 20$:
                   5B    6B    C0  0CB2  1840          ADDL2   (R11),R11                 ; Next unit block
                00000638'8F 5B    D1  0CB5  1841          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                      4E    13  0CBC  1842          BEQL    END_UPDATE                ; BR if yes
                                  0CBE  1843          $GET    RAB = (R10)               ; Get a record
                      24 50    E9  0CC7  1844          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                   C014'CF 20    8A  0CCA  1845          BICB2   #LC_BITM,BUFFER           ; Convert to uppercase
                0014'CF 55 8F   91  0CCF  1846          CMPB    #^A7U/,BUFFER             ; Is it a UCB record?
                      35    12  0CD5  1847          BNEQ    END_UPDATE                ; BR if not
                      01    E0  0CD7  1848          BBS     #UETUNT$V_TESTABLE,-      ; BR if this unit is testable...
                   D6 0B AB       0CD9  1849                  UETUNT$B_FLAGS(R11),20$
                0018'CF 4E 8F   90  0CDC  1850          MOVB    #^A/N/,BUFFER+4           ; ...else disable the UCB record...
                                  0CE2  1851          $UPDATE RAB = (R10)               ; ...here
                      C4 50    E8  0CEB  1852          BLBS    R0,20$                    ; Look at the next record if no error
                                  0CEE  1853 UPDATE_FAILED:
                   0C AA    DD  0CEE  1854          PUSHL   RAB$L_STV(R10)            ; Do a simple message...
                      50    DD  0CF1  1855          PUSHL   R0                        ; ...to tell of the failure
                01B8'CF       DF  0CF3  1856          PUSHAL  INIDEV_UPDERR
                      01    DD  0CF7  1857          PUSHL   #1
                      00    EF  0CF9  1858          EXTZV   #STS$V_SEVERITY,-         ; Copy the severity from RMS status...
                   7E 50    03        0CFB  1859                  #STS$S_SEVERITY,R0,-(SP)
             6E 00741130    8F   C8  0CFE  1860          BISL2   #UETP$_TEXT,(SP)          ; ...to our message
     00000000'GF    05    FB  0D05  1861          CALLS   #5,G^LIB$SIGNAL
                                  0D0C  1862 END_UPDATE:
                      00    DD  0D0C  1863          PUSHL   #0                        ; Set the time flag
                000F'CF       DF  0D0E  1864          PUSHAL  TEST_NAME                 ; Push the test name
                      02    DD  0D12  1865          PUSHL   #2                        ; Push arg count
                      00    EF  0D14  1866          EXTZV   #STS$V_SEVERITY,-         ; Push the proper exit severity...
                      03        0D16  1867                  #STS$S_SEVERITY,-
                   7E 0146'CF       0D17  1868                  STATUS,-(SP)
             6E 00741080    8F   C8  0D1B  1869          BISL2   #UETP$_ENDEDD,(SP)        ; ...and use it in our message code
                      04    DD  0D22  1870          PUSHL   #4
                   51 5E    D0  0D24  1871          MOVL    SP,R1
                                  0D27  1872          $PUTMSG_S MSGVEC = (R1)           ; Output the message
                                  0D36  1873          $SETPRN_S PRCNAM = ACNT_NAME      ; Reset the process name
                         04       0D41  1874          RET                               ; That's all folks!
                                  0D42  1875
```

```
                0D42  1876          .END    UETCOMS00
```

```
$$.TAB                          = 00000818 R      03
$$.TABEND                       = 0000085C R      03
$$.TMP                          = 00000000
$$.TMP1                         = 00000001
$$.TMP2                         = 0000006A
$$.TMPX                         = 00000016 R      04
$$.TMPX1                        = 0000000D
$$T1                            = 00000001
$$T2                            = 00000006
ACNT_NAME                         00000000 R      02
ALL_SET                           000003E9 R      05
ARG_COUNT                         00000182 R      03
ASTPAR_ERRMSG                     00000293 R      02
ATTN_DELM                       = 00000040
ATTN_DELV                       = 00000006
ATTN_MBX_MSG                      0000050B R      02
ATTN_MBX_TEST                     00000528 R      05
ATTN_MBX_TYPES                    00000554 R      02
ATTN_MBX_TYPES_ATTN               0000057E R      02
ATTN_MBX_TYPES_DATAVL             00000570 R      02
ATTN_MBX_TYPES_LENGTH           = 00000008
ATTN_MBX_TYPES_NAMES              0000055C R      02
ATTN_MBX_TYPES_SHUTDN             00000577 R      02
ATTN_MBX_TYPES_UNKNOWN            00000583 R      02
BAD_DATA                          00000635 R      03
BEGIN_MSGM                      = 00000008
BEGIN_MSGV                      = 00000003
BUFFER                            00000014 R      03
BUFFER_PTR                        0000000C R      03
CCASTHAND                         00000B12 R      05
CHECK_IOSB                        00000768 R      05
CHF$L_SIGARGLST                 = 00000004
CHF$L_SIG_ARG1                  = 00000008
CHF$L_SIG_ARGS                  = 00000000
CHF$L_SIG_NAME                  = 00000004
CHK_MBX_AST                       0000080D R      05
CHK_QIO_AST                       0000077F R      05
CLEAN_EXIT                        000005C8 R      05
CNTRLCMSG                         000000A3 R      02
COMMON                            00000AD9 R      05
CONTROLLER                        00000031 R      02
CONT_DESC                         000001F5 R      02
CS1                               00000082 R      02
CS3                               00000094 R      02
DDB_RFA                           00000770 R      03
DEAD_CTRLNAME                     000000E4 R      02
DEV$V_TRM                       = 00000002
DEVCHAR_BLK                       00000199 R      03
DEVDEP_SIZE                     = 00000000
DEVDSC                            00000098 R      03
DEVNAM_LEN                        00000164 R      03
DEV_NAME                          000000B7 R      03
DIB                               000000C6 R      03
DIB$B_DEVCLASS                  = 00000004
DIB$B_DEVTYPE                   = 00000005
DIB$K_LENGTH                    = 00000074
DIBBUF                            000000CE R      03
DUMMY_FAB                         000007C8 R      03
DUMMY_RAB                         00000818 R      03
DVI$_DEVNAM                     = 00000020
EFN2                            = 00000004
EF_MASK                           000001A1 R      03
END_UPDATE                        00000D0C R      05
ERROR_COUNT                       00000142 R      03
ERROR_EXIT                        00000B70 R      05
ERR_ATTN_MSG                      000004D8 R      02
ERR_FATAL_MSG                     00000304 R      02
ERR_LOST_MSG                      0000033A R      02
ERR_MAINT_MSG                     000003AE R      02
ERR_START_MSG                     0000037C R      02
ESC                             = 0000001B
EXIT_DESC                         00000172 R      03
EXIT_HANDLER                      00000C17 R      05
FAB$B_BID                       = 00000000
FAB$B_FNS                       = 00000034
FAB$C_BID                       = 00000003
FAB$C_BLN                       = 00000050
FAB$C_SEQ                       = 00000000
FAB$C_VAR                       = 00000002
FAB$L_ALQ                       = 00000010
FAB$L_DEV                       = 00000040
FAB$L_FNA                       = 0000002C
FAB$L_FOP                       = 00000004
FAB$L_STS                       = 00000008
FAB$L_STV                       = 0000000C
FAB$V_CHAN_MODE                 = 00000002
FAB$V_CR                        = 00000001
FAB$V_FILE_MODE                 = 00000004
FAB$V_GET                       = 00000001
FAB$V_LNM_MODE                  = 00000000
FAB$V_PUT                       = 00000000
FAB$V_UFO                       = 00000011
FAB$V_UPD                       = 00000003
FAB$V_UPI                       = 00000006
FAB$W_GBC                       = 00000048
FAO_BUF                           00000004 R      03
FILE                              000001FD R      02
FIND_IT                           000001E1 R      05
FLAG                              00000002 R      03
FLAG_SHUTDNM                    = 00000040
FLAG_SHUTDNV                    = 00000006
FOUND_IT                          00000279 R      05
GOOD_DATA                         00000636 R      03
ILLEGAL_REC                       00000151 R      02
INADDRESS                         00000152 R      03
INIDEV_UPDERR                     000001B8 R      02
INI_FAB                           000006DC R      03
INI_RAB                           0000072C R      03
INPUT_ITMLST                      00000072 R      02
IO$M_ATTNAST                      ********    X   05
IO$M_CTRLCAST                     ********    X   05
IO$M_SHUTDOWN                     ********    X   05
IO$M_STARTUP                      ********    X   05
IO$_READVBLK                      ********    X   05
```

```
IO$_SETMODE                     ********    X   05
IO$_WRITEVBLK                   ********    X   05
ITERATION                       00000166 R      03
LC_BITM                       = 00000020
LIB$SIGNAL                      ********    X   05
LIMIT                         = 00000010
MAX_DEV_DESIG                 = 0000000A
MAX_MSG_LEN                   = 00000200
MAX_PROC_NAME                 = 0000000F
MAX_UNIT_DESIG                = 00000005
MBXAST_DELM                   = 00000080
MBXAST_DELV                   = 00000007
MBXCHAN                         00000186 R      03
MBXLOGNAM                       00000190 R      03
MBXSIZE                       = 00000080
MBX_BUF                         000005B5 R      03
MBX_ERRMSG                      000002D0 R      02
MBX_LOGNAMSIZ                 = 00000007
MODE                            00000041 R      02
MODE_IS_ONEM                  = 00000010
MODE_IS_ONEV                  = 00000004
MSG$_XM_ATTN                  = 0000000D
MSG$_XM_DATAVL                = 0000000B
MSG$_XM_SHUTDN                = 0000000C
MSG_BLOCK                       0000016E R      03
NAME_LEN                      = 0000000F
NEW_NODE                        00000640 R      03
NOUNIT_SELECTED                 0000012B R      02
NO_CTRLNAME                     000000C4 R      02
NO_RMS_AST_TABLE                0000004D R      02
NO_WAIT_READ                    000004A3 R      02
NRAT_LENGTH                   = 00000014
ONEMIN                          000001E5 R      02
OTS$CVT_TI_L                    ********    X   05
OUTADDRESS                      0000015A R      03
PAGES                         = 00000001
PASS                            0000016A R      03
PASS_MSG                        00000185 R      02
PMTSIZ                        = 00000019
PRM                           = 00000064
PROCESS_NAME                    000000A0 R      03
PROCESS_NAME_FREE             = 0000000B
PROC_CONT_NAME                  0000008B R      05
PROMPT                          00000238 R      02
QUAD_STATUS                     0000014A R      03
RAB$B_PSZ                     = 00000034
RAB$B_RAC                     = 0000001E
RAB$C_BID                     = 00000001
RAB$C_BLN                     = 00000044
RAB$C_RFA                     = 00000002
RAB$C_SEQ                     = 00000000
RAB$L_CTX                     = 00000018
RAB$L_FAB                     = 0000003C
RAB$L_PBF                     = 00000030
RAB$L_ROP                     = 00000004
RAB$L_STS                     = 00000008
RAB$L_STV                     = 0000000C
RAB$V_PMT                     = 0000001E
RAB$W_RFA                     = 00000010
RAB$W_RSZ                     = 00000022
READ_SIZE                     = 00000000
RECORD                          00000209 R      02
RECV_AST                        000007A5 R      05
RECV_BUF                        000003B5 R      03
RECV_EFN                      = 00000008
RECV_ERR_MSG                    00000251 R      02
RECV_IOSB                       000001AD R      03
REC_SIZE                      = 00000028
RMS$_BLN                        ********    X   02
RMS$_BUSY                       ********    X   02
RMS$_CDA                        ********    X   02
RMS$_FAB                        ********    X   02
RMS$_FACILITY                 = 00000001
RMS$_RAB                        ********    X   02
RMS_ERROR                       00000AA3 R      05
RMS_ERR_STRING                  00000217 R      02
RW_TIME_ID                    = 00000003
SAFE_TO_UPDM                  = 00000004
SAFE_TO_UPDV                  = 00000002
SEC$M_EXPREG                    ********    X   05
SEC$M_GBL                       ********    X   05
SHR$_ABENDD                   = 000010E0
SHR$_BEGIND                   = 00001038
SHR$_ENDEDD                   = 00001080
SHR$_OPENIN                   = 00001098
SHR$_TEXT                     = 00001130
SS$_BADPARAM                  = 00000014
SS$_CONTROLC                  = 00000651
SS$_NORMAL                    = 00000001
SS$_NOSUCHSEC                 = 00000978
SS$_SSFAIL                    = 0000045C
SS$_TIMEOUT                   = 0000022C
SS$_WASSET                    = 00000009
SSERROR                         000009C0 R      05
SS_SYNCH_EFN                  = 00000003
START_DEV                       000006AD R      05
START_TEST                      0000040A R      05
STATUS                          00000146 R      03
STR$UPCASE                      ********    X   05
STS$K_ERROR                   = 00000002
STS$K_INFO                    = 00000003
STS$K_SUCCESS                 = 00000001
STS$K_WARNING                 = 00000000
STS$M_INHIB_MSG               = 10000000
STS$S_FAC_NO                  = 0000000C
STS$S_SEVERITY                = 00000003
STS$V_FAC_NO                  = 00000010
STS$V_SEVERITY                = 00000000
STS_DCHK_MSG                    00000424 R      02
STS_DISC_MSG                    0000046E R      02
STS_ORUN_MSG                    000003E6 R      02
STS_TIMO_MSG                    00000459 R      02
SUC_EXIT                        00000622 R      05
SUPDEV_GBLSEC                   00000020 R      02
```

```
SUP_FAB                         00000778 R      03
SYS$ASSIGN                      ******** GX     05
SYS$CANTIM                      ******** GX     05
SYS$CLREF                       ******** GX     05
SYS$CONNECT                     ******** GX     05
SYS$CREMBX                      ******** GX     05
SYS$CRMPSC                      ******** GX     05
SYS$DCLEXH                      ******** GX     05
SYS$EXIT                        ******** GX     05
SYS$EXPREG                      ******** GX     05
SYS$FAO                         ******** X      05
SYS$GET                         ******** GX     05
SYS$GETDEV                      ******** GX     05
SYS$GETDVI                      ******** GX     05
SYS$GETMSG                      ******** GX     05
SYS$INPUT                       00000061 R      02
SYS$MGBLSC                      ******** GX     05
SYS$OPEN                        ******** GX     05
SYS$PUTMSG                      ******** GX     05
SYS$QIO                         ******** GX     05
SYS$QIOW                        ******** GX     05
SYS$SETAST                      ******** GX     05
SYS$SETEF                       ******** GX     05
SYS$SETIMR                      ******** GX     05
SYS$SETPRN                      ******** GX     05
SYS$SETSFM                      ******** GX     05
SYS$TRNLOG                      ******** GX     05
SYS$UPDATE                      ******** GX     05
SYS$WAITFR                      ******** GX     05
SYS$WFLAND                      ******** GX     05
SYSIN_FAB                       00000648 R      03
SYSIN_RAB                       00000698 R      03
TEST_ERRM                     = 00000020
TEST_ERRV                     = 00000005
TEST_NAME                       0000000F R      02
TEST_OVERM                    = 00000002
TEST_OVERV                    = 00000001
TEXT_BUFFER                   = 00000084
THREEMIN                        000001DD R      02
TIME_ERR_OUT                    000009A6 R      05
TIME_ID_1                     = 00000001
TIME_ID_2                     = 00000002
TIME_SUC_OUT                    000009B8 R      05
TTCHAN                          00000000 R      03
UETCOMS00                       00000000 RG     05
UETP                          = 00740000
UETP$_ABENDD                  = 007410E0
UETP$_ABORTC                  = 0074832B
UETP$_BEGIND                  = 00741038
UETP$_DENOSU                  = 00748333
UETP$_DEUNUS                  = 0074819A
UETP$_ENDEDD                  = 00741080
UETP$_ERBOXPROC               = 00748020
UETP$_FACILITY                = 00000074
UETP$_OPENIN                  = 00741098
UETP$_TEXT                    = 00741130
UETUNT$B_FLAGS                = 0000000B
UETUNT$B_TYPE                 = 00000008
UETUNT$C_FAB                  = 00000110
UETUNT$C_INDSIZ               = 000001A4
UETUNT$K_FAB                  = 00000110
UETUNT$K_RAB                  = 00000160
UETUNT$M_TESTABLE             = 00000002
UETUNT$T_FILSPC               = 00000014
UETUNT$V_TESTABLE             = 00000001
UETUNT$W_SIZE                 = 00000009
UNIT_DESC                       000001ED R      02
UNIT_LIST                       00000638 R      03
UNIT_LOOP                       00000C89 R      05
UNIT_NUMBER                     00000162 R      03
UPDATE_FAILED                   00000CEE R      05
WRITE_SIZE                    = 00000000
XM$M_CHR_LOOPB                = 00000002
XM$M_CHR_MBX                  = 00000010
XM$V_ERR_FATAL                = 00000010
XM$V_ERR_LOST                 = 00000014
XM$V_ERR_MAINT                = 00000013
XM$V_ERR_START                = 00000017
XM$V_STS_ACTIVE               = 0000000B
XM$V_STS_DCHK                 = 00000008
XM$V_STS_DISC                 = 0000000E
XM$V_STS_ORUN                 = 0000000A
XM$V_STS_TIMO                 = 00000009
XMIT_BUF                        000001B5 R      03
XMIT_EFN                      = 00000005
XMIT_RECV                       0000047B R      05
XMMBX_DESC                      00000188 R      03
XM_ATTN_AST                     0000086A R      05
XM_CHAN                         00000197 R      03
XM_IOSB                         000001A5 R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	0000058B (1419.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	0000085C (2140.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000023 (35.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
COMS	00000D42 (3394.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.07	00:00:00.42
Command processing	115	00:00:00.70	00:00:04.76
Pass 1	543	00:00:24.11	00:00:48.82
Symbol table sort	0	00:00:02.26	00:00:03.83
Pass 2	611	00:00:06.70	00:00:16.57
Symbol table output	40	00:00:00.32	00:00:00.78
Psect synopsis output	6	00:00:00.05	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1345	00:00:34.21	00:01:15.23

The working set limit was 900 pages.
134408 bytes (263 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1540 non-local and 54 local symbols.
1876 source lines were read in Pass 1, producing 41 object records in Pass 2.
63 pages of virtual memory were used to define 56 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[UETP.OBJ]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	51
TOTALS (all libraries)	53

1868 GETS were required to define 53 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETCOMS00/OBJ=OBJ$:UETCOMS00 MSRC$:UETCOMS00/UPDATE=(ENH$:UETCOMS00)+EXECML$/LIB+LIB$:UETP/LIB
```

SATSSF18
LIS

SATSSS09
LIS

UETCOMS00
LIS

UETDISK00
LIS

SATSSS10
LIS

UETDMPF00
LIS